4. NÁVRH MIKROPOČÍTAČOVÉHO SYSTÉMU

Návrh mikropočítačového systému spočívá obvykle v návrhu a vývoji mikropočítače pro specifickou aplikaci. Základní bloky, které obsahuje mikropočítač jsou: mikroprocesor, permanentní paměť, paměť s libovolným výběrem, vstupní a výstupní zařízení a vazební obvody (styková zařízení).

Konstruktér systému kromě vývoje technického vybavení mikropočítače musí navrhnout programové vybavení, které se obvykle skládá z řídicího programu (operačního systému) a uživatelského programu. Řídící program řídí mikropočítačový systém, uživatelský program řeší specifickou úlohu.

Mikropočítač často neslouží pro obecné použití, ale pro určitou aplikaci nebo třídu aplikací. Základní problémy, které musí řešit konstruktér systému, lze rozdělit do následujících kroků:
- rozbor úlohy a stanovení kritérií systému
- určení vhodnosti použití mikroprocesoru
- návrh nestandardního technického vybavení
- vývoj uživatelského programu
- zajištění integrity technického a programového vybavení
- ladění a zkoušení
- zajištění prostředků pro údržbu.

S některými těmito kroky jsme se již zabývali. Uveďme si několik poznámek k ostatním, přičemž programování, vzhledem k jeho důležitosti, bude věnována zvláštní pozornost.

4.1 Kritéria systému

Prvním krokem při návrhu mikropočítačového systému je stanovení souboru kritérií, která se bezprostředně vážou k zamýšlené aplikaci. Mezi ně obvykle patří: cena, pružnost, slučitelnost, spolehlivost, rychlost, velikost.

Cena je obvykle dominujícím kriteriem; bohužel je obtížné ji stanovit předem, protože při řešení prototypu jsou náklady spojené s návrhem systému a programováním podstatně vyšší než náklady na technické vybavení.

Pružnost mikropočítačového systému, promítající se zpravidla jen ve výměně permanentní paměti, ve které je zapsán program, je obvykle značnou výhodou s ohledem na případnou změnu požadavků nebo inovaci.

4.1.1 Vhodnost použití mikroprocesoru

Po stanovení kritérií systému musí konstruktér rozhodnout, zda je vhodný systém s mikropočítačem. V tomto kroku musí uvažovat strukturu technického a programového vybavení a systémovou integritu.

Struktura technického vybavení se vztahuje k specifikaci mikroprocesoru; je třeba uvažovat napájecí napětí, kmitočet a fáze zdroje hodinových impulsů, příkon atd. Obvykle tyto záležitosti nejsou podstatné.

Struktura programového vybavení se vztahuje na operační vlastnosti mikroprocesoru, např. počet a typy registrů, délku slova, způsoby adresování, počet a typy instrukcí, maximální kapacitu paměti. Tyto otázky jsou obvykle zásadní při každé aplikaci.

Důležitým faktorem je systémová integrita. Mikroprocesor musí být spojen s peramentní pamětí, s pamětí s libovolným výběrem, se vstupy a výstupy a jinými logickými obvody. Slučitelnost mezi těmito uzly musí být jak na úrovni technického vybavení (např. napájení, hodiny), tak na úrovni programového vybavení (např. délka slova, tvar slova).

4.1.2 Prostředky pro údržbu

Důležitým aspektem spolehlivého návrhu mikropočítačového systému je vývoj vhodných prostředků pro údržbu. Je celá řada možností zajišťující údržbu systému. Pravděpodobně nejsnadnější způsob je použití diagnostického programu, který kontroluje funkci jednotlivých uzlů mikropočítačového systému a vhodně indikuje případné poruchy. Tento diagnostický program může být zapsán v permanentní

paměti, která při zkoušení nahrazuje v mikropočítačovém systému permanentní paměť zajišťující normální funkci.

Jinou metodou je použití podprogramů detekujících chybu, které jsou vyvolávány operačním systémem.

Umožnění diagnostiky jak technického tak programového vybavení mikropočítačového systému je důležitým faktorem při celkovém návrhu systému.

4.2 Postup při programování mikropočítače

Programování mikropočítače je jeden z nejnáročnějších kroků při návrhu systému. Je jen málo případů, kdy výrobce technického vybavení dodává všechny programy. Většinou uživatel mikropočítače si musí sestavit alespoň uživatelský program pro svou úlohu. Různé fáze rozhodnutí, které je třeba provést během programování, jsou uvedeny na obr. 4.1.

Činnost konstruktéra - programátora můžeme rozložit do následujících kroků:

- definice systému
- zvolení algoritmu (stanovení postupu sestavy programu)
- kódování (zápis programu ve zvoleném jazyku)
- překlad
- zkoušení a ladění

4.2.1 Definice systému

Dříve než se přistoupí k vlastnímu programování, je nezbytné specifikovat jednotlivé uzly, z kterých se skládá mikropočítačový systém. Tato specifikace musí alespoň obsahovat:

- určení vstupů a výstupů
- určení paměti
- rozbor aritmetických a logických operací

Definice vstupů a výstupů

V souvislosti s programováním nás u vstupních a výstupních zařízení především zajímá kódování dat s cílem slučitelnosti s mikropočítačovým systémem. Rovněž zajištění synchronismu mezi přídavnými zařízeními a mikroprocesorem není druhořadou záležitostí.

Znaky pro přenos a záznam je třeba zakódovat. Obvykle se kódu-

je: 26 abecedních znaků (velká písmena), 26 abecedních znaků (malá písmena), 10 číslicových znaků a často přibližně 25 různých značek včetně dálnopisných a jiných přenosových informací. Je třeba tedy zakódovat přibližně 90 znaků, odkud plyne nezbytnost použití 7 bitů ($2^7 = 128$).

Byla vypracována řada kódů pomocí 6, 7 a 8 bitů pro kódování abecedně číslicových znaků; některé nekódují malá písmena. Nejznámější a nejčastěji používané kódy jsou ASCII (American Standard Code for Information Interchange); tab. 4.1 a EBCDIC (Extended Binary CODED Decimal Interchange Code); tab. 4.2. Pro kódování latinky a azbuky se často používá kód DKOI (tab. 4.3).

Při aplikacích v komunikacích je třeba kromě kódování dat ještě tato data uspořádat do skupin tak, aby terminál nebo komunikační zařízení vědělo, kdy se data přenášejí a kdy zpráva začíná nebo končí. Tyto způsoby se nazývají „řízení spojování dat" (date link control nebo line protocol); dosud nejsou mezinárodně normalizovány. Různé firmy však používají „své" standardy, např. IBM používá SDLC (Synchronous Date Link Control), firma Digital Equipment Corporation používá DDCMP (Digital Date Communication Message Protocol).

Paměti

Při volbě operační paměti je třeba uvažovat typ permanentní paměti (programovatelná maskou, elektricky a reprogramovatelná), zapojení čipu i uspořádání celé paměti, typ paměti s libovolným výběrem a její chování při přerušení napájení.

Aritmetické a logické operace

Aritmetické a logické operace je třeba považovat za klíčové z hlediska programování. Různé typy mikroprocesorů je řeší s různou rychlostí. Při návrhu systému je obvykle účelné odhadnout typy instrukcí a též četnost jejich výskytu v uživatelském programu. Z těchto informací a případně z požadavku na rychlost se může provést volba mikroprocesoru. Např. předpokládají-li se často aritmetické výpočty v desítkové soustavě, je žádoucí, aby soubor instrukcí mikroprocesoru obsahoval instrukce pro aritmetické výpočty v dvojkově desítkové soustavě.

Existují tři základní způsoby, kterými se provádějí aritmetic-

ké a logické operace: použití instrukcí mikroprocesoru, mikroprogramování a neměnně zapojený logický obvod. Podstatný rozdíl mezi těmito přístupy je v rychlosti. Ve většině případů vyhovuje rychlost, kterou umožňuje mikroprocesor svými instrukcemi. V některých aplikacích, nejčastěji u systémů pracujících v reálném čase se rychlost stává kritická. Řešení může poskytnout mikroprogramování, pokud tuto techniku mikroprocesor umožňuje.

Neměnně zapojený logický obvod provádějící určité aritmetické nebo logické operace nelze též předem vyloučit z úvah o mikropočítačovém systému, zvláště je-li rychlost kritická a tyto operace se periodicky opakují. Někdy je právě účelné, aby tento neměnně zapojený logický obvod byl součástí mikropočítačového systému, neboť za cenu částečného rozšíření technického vybavení se podstatně zrychlí výpočet. Příklad, kdy se periodicky opakují aritmetické operace čas je mnohdy kritický, jsou číslicové filtry. Zde je někdy účelné použít neměnně zapojeného logického obvodu - násobičky, která několikanásobně urychlí výpočet a může být jediným řešením umožňujícím činnost v reálném čase.

4.3 Prostředky k sestavení programu

Stanovení postupu pro sestavení programu je proces, ve kterém celou úlohu rozložíme do jednotlivých dílčích kroků. Obvykle je výsledkem této etapy návrhu detailní vývojový diagram, někdy doplněný popisem programovací strategie. Během této fáze se konstruktér obvykle snaží minimalizovat potřebnou kapacitu paměti. Pokud má mikropočítač řešit úlohu v reálném čase, je nezbytné v této fázi přihlížet k době řešení jednotlivých částí úlohy.

Vývojový diagram umožňuje rozložit řešenou úlohu do dílčích kroků a znázornit jednotlivé vazby mezi těmito dílčími kroky. Je to v podstatě schématické znázornění nezbytných kroků pro řešení úlohy. Jednotlivé bloky různých tvarů ve vývojovém diagramu představují různé operace. Spojnice mezi bloky znázorňují sled operací a jsou často doplněné šipkami pro označení jejich sledu.

Vývojový diagram je užitečný nejen pro sestavu programu, ale též pro odstranění chyb v napsaném programu. Výhoda použití vývojového diagramu spočívá v názorném zobrazení logického postupu při řešení úlohy. Při výkladu či popisu programu je použití vývojového

diagramu podstatně přehlednější než sledování dlouhého sledu instrukcí.

Vývojový diagram pro určitou úlohu může být sestaven na různé úrovni. Obvykle se sestavuje při řešení složitější úlohy základní (hrubý) vývojový diagram, někdy nazývaný funkční postupový diagram. Vyjadřuje požadovanou funkci systému jednotlivých větších funkčních krocích.

Po kontrole funkce základního postupového diagramu následuje vypracování podrobnějšího vývojového diagramu, který někdy nazýváme programovací vývojový diagram. Jeho jednotlivé kroky vyjadřují instrukce mikroprocesoru nebo se jim alespoň přibližují. Např. označuje-li jeden z kroků funkčního postupového diagramu „vynuluj systém", může být tento krok rozepsán v programovacím vývojovém diagramu do dvou kroků: „vynuluj registr A" a „vynuluj registr B".

Funkční postupový diagram je zpravidla sestaven bez ohledu na konkrétní typ mikroprocesoru. Programový vývojový diagram je při programování v jazyku symbolických adres závislý na konkrétním typu mikroprocesoru, při programování ve vyšším programovacím jazyku je strojově nezávislý.

Úroveň zpracování vývojového diagramu závisí na programátorovi a řešené úloze. Dva programátoři mohou sestavit odlišné jak funkční tak programovací vývojové diagramy pro stejnou úlohu.

Při sestavě vývojového diagramu používáme některých obecných symbolů, které jsou určeny československou státní normou ČSN 369030 (Značky vývojových diagramů pro systémy zpracování informací). Nutno však podotknout, že v detailech vývojového diagramu se může projevit osobitý zápis programátora. Obdélníkem znázorňujeme provedení jakékoliv operace (skupiny operací), jejímž výsledkem je transformace informace (např. změna hodnoty, tvaru, umístění, poslání). Dovnitř obdélníka zaznamenáváme příslušnou operaci. Značka pro zpracování má obvykle jeden vstup a pouze jeden výstup.

Kosočtvercem znázorňujeme rozhodování, větvení, přepínání. Představuje rozhodovací nebo přepínací typ operace, která určuje alternativní cestu (větvení) dalšího vývoje diagramu. Symbol pro rozhodování, větvení, přepínání někdy též nazýváme rozhodovací blok; má obvykle jeden vstup a dva nebo tři výstupy. Při dvou výstupech

často u jednoho výstupu píšeme ano, u druhého ne, čímž označíme sled operací při splnění nebo nesplnění uvedených podmínek.

Pro označení návaznosti vývojového diagramu (např. při detailování základního vývojového diagramu nebo dlouhém vývojovém diagramu) používáme spojky. Značí přechod na jinou část nebo z jiné části vývojového diagramu. Pro mezní značku používáme ovál; značí mezní bod vývojového diagramu, t.j. začátek, konec, zastavení. Nejužívanější značky ve vývojovém diagramu jsou uvedeny na obr.4.2.

Příklad funkčního postupového diagramu je na obr. 4.3, který vyhledává přídavná zařízení, označuje získaná data a ukládá je do paměti. Program příslušející uvedenému funkčnímu postupovému diagramu je též příkladem hodnotícího programu (Benchmark program). Cílem hodnotícího programu je vyhodnotit určitý typ mikroprocesoru pro určitou úlohu (nebo třídu úloh), přičemž kritériem je potřebná kapacita paměti pro uložení programu a čas potřebný pro provedení úlohy. Relativní důležitost rychlosti či kapacity závisí na konkrétní aplikaci.

Jako další příklad si uveďme vývojový diagram úlohy, která spočívá ve stanovení největšího čísla ze souboru čísel a_1, a_2, ... a_n. Zjednodušme si nejprve tuto obecnou úlohu. Určeme největší číslo ze tří čísel (t.j. n=3). Vývojový diagram pro tuto úlohu je na obr. 4.4. Pro usnadnění řešení úlohy jsme si zavedli proměnnou z. Prvním přikazovacím blokem volíme $z = a_1$, t.j. paměťovou buňku z naplníme obsahem paměťové buňky a_1. V druhém kroku, t.j. v prvním rozhodovacím bloku zkoumáme, zda $z = a_1$ je větší či menší než a_2. Je-li splněna podmínka $z = a_1 < a_2$, volíme $z = a_2$, t.j. do paměťové buňky z, kde bylo zapsáno číslo a_1 zapíšeme nové číslo a_2. Není-li splněna podmínka $a_1 < a_2$, t.j. je-li číslo a_1 větší než číslo a_2, zůstává $z = a_1$. V další části vývojového diagramu porovnáváme větší z čísel a_1, a_2 s číslem a_3.

Při porovnávání více čísel ($n > 3$), můžeme „prodloužit" uvedený vývojový diagram. Jiný přístup je na obr. 4.5. Proměnná j postupně narůstá až do hodnoty n, čímž je postupně porovnáváme dvojice čísel ze souboru čísel a_1, a_2, ... a_n a vyhledáváme největší číslo. V druhém přikazovacím bloku volíme $j = 2$. Použití proměnné j má dvojí účel: jednak nám umožňuje postupně porovnávat další čísla ze souboru a_j ($j = 2, 3, ...n$), jednak umožňuje indikaci vyčer-

pání všech porovnávaných čísel. První rozhodovací blok ve vývojovém diagramu zkoumá, zda proměnná j je menší či větší než počet porovnávaných čísel n, t.j. zda jsme již vyčerpali celý soubor zkoumaných čísel a_1, a_2, ... a_n a tím dokončili úlohu. Poslední přikazovací blok (j ← j +1) zvyšuje proměnnou j o jedničku, abychom mohli postupně porovnávat všechna čísla ze souboru a_1, a_2 až a_n.

Poznamenejme, že jednotlivé bloky ve vývojovém diagramu nemusí odpovídat instrukcím v určitém jazyku. Pro ilustraci jsou v pravé části obou předcházejících vývojových diagramů uvedeny příkazy v jazyku FORTRAN; zpravidla se však do vývojového diagramu program nepíše. Jazyk FORTRAN používá jen velká písmena, nepoužívá indexů ani symbolů <, > , ← atd. Hodnotu a_1 zapisujeme A(1) (index zapisujeme jako číslo v kulaté závorce následující za písmenem). Symbol < zapisujeme LT (zkratka z less than, menší než). Symbol ← (nahraď) zapisujeme znakem =, který však neznamená rovnost.

První, druhý a pátý řádek programu na obr. 4.5 jsou přiřazovací příkazy, které mají za úkol přiřadit proměnným hodnoty (které mohou být výsledkem výpočtu aritmetického výrazu). Např. Z = A(1) značí přiřadit proměnné z hodnotu a_1, t.j. přenést obsah paměťové buňky a_1 do paměťové buňky z. Podmíněný příkaz IF (J.GT.N) GO TO 6 značí „je-li j větší než n, pokračuj na návěští 6 v opačném případě pokračuj na dalším příkazu". Podobně podmíněný příkaz IF(Z.LT.A (J)) značí „je-li z menší než a_j, přiřaď proměnné hodnotu z hodnotu a_j; není-li z menší než a_j, proveď další příkaz". Řídicí příkaz GO TO 1 značí „jdi na návěští 1", t.j. proveď příkaz s návěštím 1. Instrukce STOP označuje logický konec procedury, END označuje konec programu.

4.4 Kódování

Proces, při kterém z programovacího vývojového diagramu sestavujeme program, nazýváme kódování. Jinými slovy kódováním označujeme činnost, při které postupně programovací vývojový diagram převádíme na instrukce mikroprocesoru. Kódování v jazyku symbolických adres v podstatě spočívá v postupném vyhledávání vhodných operačních znaků ze souboru instrukcí mikroprocesoru a určování operandů. V tomto jazyku obsahuje kódovací tabulka obvykle následující pole:

návěští operační znak operand komentář

Návěští vyjadřuje určité místo v paměti pro uložení příslušné instrukce. Vyjádření paměťového místa není obvykle absolutní, ale je označeno jen symbolicky. Návěštím označujeme všechny instrukce, ale jen ty, na které provádíme odkazy, např. uskutečňujeme skok.

Operační znak mnemotechnicky vyjadřuje strojovou instrukci. Může též obsahovat informaci o způsobu adresování nebo jiné skutečnosti ovlivňující provedení instrukce. Operace vyjádřená operačním znakem se provádí s operandem, který je specifikován buď přímo nebo adresou. Pole komentáře je určeno pro poznámky programátora; při překladu programu se neuvažuje.

4.5 Zkoušení a ladění

Zkoušení a ladění je poslední krok při vývoji programu. Zkoušení je činnost zaměřená na odhalení chyb způsobených chybným návrhem programu, kódováním a v případě aplikací vyžadujících činnost v reálném čase chybným časováním. Seznam potenciálních zdrojů chyb je v tab. 4.4. Jediná cesta, kterou zjistíme bezchybnost programu, je jeho zavedení do paměti vyvíjeného mikropočítače a jeho provedení ve všech možných kombinacích vstupních hodnot. Tento přístup je proveditelný jen v případě, je-li mikropočítač vybaven vstupně-výstupním zařízením, např. dálnopisem nebo alespoň jednodušším zařízením, např. zobrazením pomocí elektroluminiscenčních polovodičových svítících diod, které jsou zapojeny na důležitých vstupech a výstupech. V tomto případě může programátor určit zkušební podmínky a sledovat, zda jsou splněny.

V případě, že tyto ideální podmínky pro zkoušení nejsou splněny, t.j. není-li skutečný systém k dispozici, pak může být předběžné zkoušení provedeno na vývojovém mikropočítačovém systému, který výrobce obvykle vyrábí pro určitý typ mikroprocesoru. Tento vývojový počítač je univerzální simulátor. Někteří vývojoví pracovníci dávají přednost dokonalejšímu vývojovému systému, který může např. sestávat z mikropočítače, který je propojen kanálem blokových přenosů s minipočítačem, u něhož využíváme především jeho větší kapacitu paměti a dokonalejší i rozsáhlejší soubor zařízení.

Poznamenejme, že program se nejsnadněji zkouší v zařízení, v němž byl odvozen, což bývá někdy mikropočítačový vývojový systém nebo větší počítač. V tomto případě je cílem zkoušení a ladění,

vyhledání a opravení chyb programu před jeho zavedením do mikropočítače, ve kterém bude používán. Jelikož program bývá často ukládán do permanentní paměti programované maskou (t.j. během výrobního procesu), je třeba jej vyzkoušet před jeho uložením.

Dále je třeba provést kontrolu přídavných zařízení; zvláštní pozornost je třeba věnovat časování a synchronizaci mezi rychlými a pomalými přídavnými zařízeními a mikroprocesorem.

Ladění je proces, při kterém odhalujeme a odstraňujeme chyby programu. Jak jsme si již uvedli, může být prováděno použitím úpravného programu (editor program) pro opravu zdrojového programu, nebo ladicího programu (debugger) pro opravu cílového programu. Ladění se též provádí programem, nazývaný simulátor.

Po úspěšném proběhnutí všech uvedených kroků je připraven bezchybný cílový program pro uvažovaný mikroprocesorový systém. Pro orientaci si uveďme doby strávené jednotlivými činnostmi při programování:

návrh programu a dokumentace	30%
kódování (jazyk symbolických adres)	20%
překlad	8%
zkoušení	12%
ladění	30%

4.6 Technika programování

Úspěšný vývoj systému s mikroprocesorem závisí na řadě faktorů, např:

- na úplném pochopení povahy i detailů uvažované aplikace
- na zdárném návrhu vazebních obvodů (stykových zařízení; interface)
- na zkušenosti v programování

Po dosažení dobrých výsledků je nezbytné volit vhodný kompromis mezi technickým a programovaným vybavením. Velmi jednoduché technické vybavení má obvykle za následek složitější a nákladnější programování. Typický tým pro vývoj systému s mikropočítačem sestává ze systémového pracovníka, konstruktéra logických obvodů, programátora a technika. Je-li však konfigurace systému minimální, může být technické i programové vybavení řešeno jedním pracovníkem.

Při návrhu je nezbytná pečlivá programová dokumentace pomocí vývojového diagramu, z kterého je zřejmá strategie programu. Tato dokumentace může být nezbytná např. pro:

- zkoušení a ladění technického i programového vybavení
- budoucí změny v uživatelském programu
- odstranění drobných chyb, které se zjistí během používání systému
- převzetí vývojových prací jiným pracovníkem.

Zkoušení a ladění může představovat 40% i více času vynaloženého na programování. Je žádoucí, aby při sestavě programu byl brán zřetel na jednoduché zkoušení a snadnou opravu programu. Bohužel i přes pečlivé zkoušení a ladění nelze zaručit, že při využívání systému se neobjeví chyby, způsobené nedostatky sestaveného programu.

Některé systémy, zvláště složitější, vyžadují diagnostický program, který může provést kontrolu správné funkce mikropočítače včetně přídavných zařízení. Proto je účelné při sestavě uživatelského programu brát zřetel na snadnou sestavu diagnostického programu.

4.6.1 Užívání instrukcí

Jak jsme si již uvedli, je programování do značné míry závislé nejen na počtu instrukcí, ale i na jejich účinnosti. Dalším důležitým faktorem je počet a funkce obecných (pracovních) registrů v mikroprocesoru a způsoby adresování. Zpravidla je účelné dávat přednost instrukcím, při kterých se informace přesouvají mezi registry před instrukcemi, které operují s operační pamětí, nebo používají okamžité (následné) adresování. Tím se použije kratší instrukce, t.j. instrukce sestávající z menšího počtu bytů. Jako příklad si uveďme požadavek vynulování střadače u Intel 8080, což můžeme provést jednou z následujících dvou instrukcí:

```
XRA A
MVI A, Ø
```

První instrukce (logická) XRA M obecně značí provést neekvivalenci (Exclusive - OR) mezi bytem v operační paměti adresovaným registry H a L a obsahem střadače, v našem případě značí neekvivalenci obsahu registru A (střadače) k téže informaci. Instrukce je jed-

nobytová. Druhá instrukce MVI M, konst. patřící do skupiny instrukcí pro přesuny, obecně značí přenést konstantu na adresu, která je určena obsahem registrů H a L. V našem případě instrukce přímo zavádí nulu do registru A, t.j. do střadače. Tato instrukce je dvoubytová. Je zřejmé, že první instrukce umožňuje kratší program.

Nezávisle na programovacím jazyku je vhodné zachovávat určitá pravidla při sestavě programu. E.Dijkstra zpozoroval, že programy lze podstatně snadněji analyzovat, jestliže se používají 3 základní typy postupů sestavy programu (obr. 4.6):
- jednoduchý sled instrukcí
- podmíněné větvení
- smyčka

Všechny tři typy mohou být vzájemně vkládány. Programy, které používají uvedené tři typy postupů mohou být poměrně snadno rozděleny do modulů. Z hlediska snadného rozdělení programu je účelné vyloučit skoky (GO TO, JUMP), které tvoří vazby mezi částmi programu.

4.6.2 Přesnost výpočtu, měřítko

Mikropočítače mají zpravidla krátké slovo dat (nejčastěji 8 bitů). Pro dosažení větší přesnosti je často třeba použít dvě slova. V tom případě se programování značně zjednoduší, obsahuje-li soubor instrukcí instrukce pro dvojnásobnou přesnost.

V mikropočítačích se obvykle uskutečňují aritmetické operace v dvojkové soustavě s pevnou řádovou čárkou, což je opodstatněno jednoduchostí technického vybavení, kratší dobou výpočtu a menší potřebnou kapacitou paměti. Krátké slovo dat (i při použití dvojnásobné přesnosti) vyžaduje vzhledem k možnosti přetečení nebo nepřesnosti výpočtu určitou opatrnost při volbě měřítka, t.j. při přiřazování vnějších číselných hodnot hodnotám uvnitř počítače. Obvykle je výhodné volit všechny číselné hodnoty X uvnitř počítače vzhledem ke strojové jednotce podle následujícího vztahu:

$$-S \leqq X \leqq S,$$

kde S je strojová jednotka, která je ekvivalentní „plnému" slovu dat. Při osmibitovém slovu dat to odpovídá přesnosti $2^{-8} = 0{,}4\%$. Obvykle při operacích v pevné řádové čárce se umísťuje řádová čárka před nejvýznačnější bit (bit s nejvyšší váhou).

Skutečné číselné hodnoty x vně mikropočítače jsou vázány k číselným hodnotám X uvnitř počítače vztahem

$$X = m_x x,$$

kde m_x je koeficient měřítka (měřítko). S cílem maximální přesnosti aritmetických operací koeficient m_x volíme tak, aby vždy $X \leqq 1$. Mnohdy je výhodné volit tento koeficient podle vztahu

$$m_x = \frac{1}{2^n} \leqq \frac{1}{\max |x|},$$

kde n je nejmenší možné celé číslo. Jako příklad si uveďme výpočet vztahu

$$y = a \sin x,$$

kde $0 \leqq a \leqq 10$, $-180^\circ \leqq x \leqq +180^\circ$

Podle předcházejícího koeficienty měřítek jsou

$$m_x = 1/2^8 = 1/256, \quad m_a = m_y = 1/2^4 = 1/16$$

Použitím těchto koeficientů předcházející vztah upravíme na tvar

$$\frac{y}{16} = \frac{a}{16} \sin\left[\left(\frac{x}{256}\right) 256\right]$$

Výrazy v kulatých závorkách jsou proměnné stroje, můžeme je označit Y, A, X. Poznamenejme, že v dvojkové soustavě násobení celistvou mocninou dvou, např. 256, znamená jen posun původního čísla o určitý počet dvojkových řádů.

5. PŘÍKLADY PROGRAMOVÁNÍ

V této kapitole si uvedeme některé jednodušší programy, které se v různých obměnách častěji vyskytují. Pozornost je především věnována základním aritmetickým operacím jak v dvojkové tak dvojkově desítkové soustavě, logickým operacím, skokům v programu a přenosu dat do podprogramu. Programy jsou psány v dosud u mikropočítačů nejužívanějším programovacím jazyku, tj. v jazyku symbolických adres. Pro ilustraci jsou též některé programy uvedeny ve strojovém jazyku.

5.1 Sčítání a odčítání v dvojkové soustavě

Uvažujme jednoduchou aritmetickou operaci

A = B + C - D

kde B, C, D jsou čísla celá. Program v jazyku symbolických adres sestavený pro mikropočítač s mikroprocesorem Intel 8080 může být následovný (viz tab. 3.3)

MVI lll	B	zavedení čísla B do registru označeného lll, což je střadač
ADI	C	přičtení čísla C k obsahu střadače
SUI	D	odečtení čísla D od obsahu střadače

Výsledek A je uložen ve střadači v druhém doplňku. Poznamenejme, že uvedený program pro požadovanou aritmetickou operaci je jeden z možných.

5.2 Vytvoření smyčky

Při programování se často používá smyčka pro iterativní výpo-

čet. Může mít tvar

```
      LXI D  A  (zavedení čísla A do registru D )

SMYCK ----  ┐
      ----  } (instrukce, které se provedou A krát)
      ----  }
      ----  ┘

      DCR D     (zmenšení obsahu registru D o jedničku)

      JNZ SMYCK (návrat na návěští SMYCK, pokud není obsah
                 registru D nulový)

      -----     (další instrukce programu)
```

Při každém průchodu smyčkou se zmenšuje obsah registru D o jedničku. Jelikož požadujeme, aby byla smyčka provedena A krát, je do registru D uloženo číslo A. Na konci smyčky se vždy testuje obsah registru D; není-li nulový, smyčka se opakuje od návěští SMYCK; je-li nulový, pokračuje výběr dalších instrukcí za instrukci JNZ SMYCK.

5.3 Násobení konstantou použitím smyčky

Mikropočítače obvykle nemají instrukci pro násobení. Tuto aritmetickou operaci můžeme však převést na postupné přičítání. Uvažujme aritmetickou operaci $y = 5 . x$. Nechť x je uloženo na adrese X a výsledek chceme uložit na adrese Y. Příslušný program pro mikropočítač s mikroprocesorem Intel 8080 MCS-8 je v tab. 5.1.

Je-li násobicí konstanta větší, je výhodné postupné přičítání provést pomocí smyčky. Uvažujme obecnou násobicí konstantu r, tj. $y = r . x$. Postupový diagram pro řešení uvedené úlohy je na obr. 5.1, ve kterém připouštíme též $r = 0$. Střadač je označen A. Při každém oběhu smyčkou jeho obsah se zvětší o x, současně se zmenší r o jedničku. Je-li $r = 0$, nastává výstup ze smyčky. Odpovídající program je v tab. 5.2, ve kterém předpokládáme, že celé číslo r je uloženo na adrese R a přesouvá se do pracovního zápisníkového registru B, jehož obsah při každém oběhu smyčkou se zmenší o jedničku.

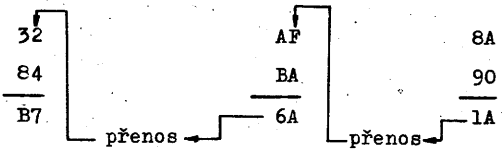
Uvedený program by bylo možné zkrátit přímým zavedením hodnoty r do registru B.

5.4 Sečtení a odečtení vícebytových čísel

U mikroprocesoru 8080 může být použit přenosový bit instrukce ADC (sečtení s přenosem) k sečtení dvou čísel libovolné délky bez znaménka. Uvažujme sečtení následujících dvou tříbytových čísel:

```
  00110010 10101111 10001010    šestnáctkově:   32AF8A
+ 10000100 10111010 10010000                  + 84BA90
-----------------------------                 ---------
  10110111 01101010 00011010                    B76A1A
```

Sečtení lze rozložit do několika kroků. Při prvním kroku se sečte nejnižší byte. V druhém kroku se přičte případně vzniklý přenos k dalším dvěma bytům podle následujícího schématu:



Příslušný program je v tab. 5.3. Registr E obsahuje délku sčítaných čísel vyjádřenou v bytech. Sčítaná čísla jsou uložena od nejnižšího do nejvyššího bytu od adresy s návěštím PRVNI a DRUHY. Výsledek součet je uložen též od nejnižšího do nejvyššího bytu počínaje adresou PRVNI. Všimněme si, že žádná instrukce, kromě ADC neovlivňuje klopný obvod příznaku přenosu. Po dosažení návěští DONE jednotlivé paměťové buňky se symbolickými adresami PRVNI až PRVNI + 2 obsahují 1A, 6A, B7, což je výsledek seřazený od nejnižšího k nejvyššímu bytu.

Při odčítání vícebytových čísel bez znaménka můžeme u mikroprocesoru 8080 použít instrukci SBB odčítání s výpůjčkou . Přenosový bit využijeme pro výpůjčku.

Uveďme si nejprve rozdíl mezi instrukcí SUB (odčítání) a instrukcí SBB. Při použití instrukce SUB se specifikovaný byte odčítá od obsahu střadače v druhém doplňku. Nedojde-li při tomto odčítání k přenosu z nejvyššího řádu, nastaví se klopný obvod příznaku

přenosu. To je opačná funkce než u operace sčítání, kdy se nastaví klopný obvod příznaku přenosu při vzniku přenosu.

Jako příklad si uveďme SUB A, což značí odečtení obsahu střadače od jeho vlastního obsahu. Předpokládejme, že střadač obsahuje $3E_{16}$

```
  0 0 1 1 1 1 1 0   = 3E16
  1 1 0 0 0 0 1 0   = druhý doplněk 3E16
 ---------------
1|0 0 0 0 0 0 0 0   = 0 /výsledek/
 ↵
přenos
```

Jelikož vznikl přenos z nejvyššího řádu, nuluje se klopný obvod příznaku přenosu, tj. přenosový bit = 0.

Při použití instrukce SBB se nejprve přičte přenosový bit k specifikovanému bytu, a potom se přičte druhý doplněk tohoto dílčího výsledku k obsahu střadače. Jako příklad si uveďme instrukci SBB L; nechť registr L obsahuje 2, střadač 4 a přenosový bit = 1

```
  1 1 1 1 1 1 0 1   = druhý doplněk 03₁₆ = druhý
                      doplněk /02₁₆ + přenos/
  0 0 0 0 0 1 0 0
 ---------------
1|0 0 0 0 0 0 0 1   = 01₁₆
 ↵
přenos
```

Jelikož vznikl přenos, nuluje se klopný obvod příznaku přenosu, neboť se jedná o odčítání.

Jako příklad odčítání vícebytových čísel bez znaménka uvažujme následující dvoubytová čísla

```
  0 0 0 1 0 0 1 1   0 0 0 0 0 0 0 1   šestnáctkově: 1 3 0 1
- 0 0 0 0 0 1 0 1   0 0 0 0 0 0 1 1                - 0 5 0 3
---------------   ---------------                 ---------
  0 0 0 0 1 1 0 1   1 1 1 1 1 1 1 0                 0 D F E
```

Nejprve se odečtou nižší byty obou čísel. Následuje odečtení vyšších bytů obou čísel při uvážení přenosového bitu (pokud vznikla výpůjčka při odčítání nižších bytů). Pro nižší byty

obdržíme

```
0 0 0 0 0 0 0 1   = 0 1(16)
1 1 1 1 1 1 0 1   = druhý doplněk 03(16)
---------------
1 1 1 1 1 1 1 0   = FE(16)
```

Jelikož nenastal přenos, nastaví se klopný obvod příznaku přenosu, který indikuje výpůjčku. Pro vyšší byte obdržíme

```
0 0 0 1 0 0 1 1   = 13(16)
1 1 1 1 1 0 1 0   = druhý doplněk 06(16) = druhý doplněk
                    (05 + přenosový bit)
---------------
0 0 0 0 1 1 0 1
```

Jelikož nastává přenos, přenosový bit = 0, což značí, že nenastává výpůjčka.

Program pro odečtení vícebytových čísel je stejný jako program pro sčítání vícebytových čísel (tab. 5.3), pouze zaměníme instrukci ADC za instrukci SBB.

5.5 Sečtení dvou čísel v dvojkově desítkovém kódu

Chtějme sečíst dvě šestnáctimístná čísla v desítkové soustavě. Za předpokladu, že délka slova počítače je alespoň 16 . 4 = 64 bitů a tento počítač umožňuje dvojkově desítkové operace, je postup výpočtu dosti jednoduchý. Je-li první sčítanec na adrese A a druhý sčítanec na adrese B a výsledek chceme uložit na adresu C, je postup výpočtu následující:

1 Přenos obsahu paměťové buňky A do střadače
2 Dvojkově desítkové přičtení obsahu paměťové buňky B k obsahu střadače
3 Přenos obsahu střadače do paměťové buňky C.

Použijeme-li mikropočítač s délkou slova 8 bitů, umístíme do jednoho slova v přirozeném dvojkově desítkovém kódu jen dvě číslice. Pro sečtení dvou šestnáctimístných čísel musíme osmkrát opakovat předcházející postup, k čemuž s výhodou můžeme použít smyčku v programu. Nechť první dvě číslice nejnižší řády prvního sčítanee X jsou uloženy na adrese PRVX a dalších čtrnáct číslic téhož sčítance na bezprostředně následujících sedmi adresách.

Podobně nechť první dvě číslice nejnižší řády druhého sčítance Y jsou na adrese DRUY a další číslice Y na bezprostředně navazujících adresách. Požadujeme uložit první dvě číslice součtu na adresu PRVX, ostatní číslice na adresy bezprostředně následující. Tím samozřejmě při výpočtu postupně mažeme první sčítanec. Počet oběhů smyčkou budeme čítat čítačem CS. Postupový diagram výpočtu je na obr. 5.2. Jednotlivé kroky značí:

1) CS = 8 (nastavení čítače CS na požadovaný počet smyček)
2) ACC←ACC + PRVX (zavedení prvních dvou číslic prvního sčítance do střadače)
3) ACC←ACC + DRUY (dvojkově desítkové přičtení prvních dvou číslic druhého sčítance k obsahu střadače)
4) PRVX←ACC (uložení obsahu střadače na adresu PRVX)
5) PRVX←PRVX + 1 (zvětšení adresy PRVX o jedničku)
6) DRUY←DRUY + 1 (zvětšení adresy DRUY o jedničku)
7) CS←CS - 1 (zmenšení obsahu čítače CS o jedničku)
8) Pro CS > 0 (návrat na 2 , CS = 0 značí konec sčítání.)

Sestavme nejprve program ve strojovém kódu pro mikroprocesor Intel 8080. Nechť první sčítanec X je uložen na adresách 100_{16} až 107_{16}, druhý sčítanec Y je uložen na adresách 108_{16} až $10F_{16}$. Pro adresu PRVX vyhraďme obecné pracovní registry D a E, pro adresu DRUY registry H a L a pro čítač oběhů smyčkou CS registr C. Sled instrukcí pro výpočet je uveden v tab. 5.4. V této tabulce značí $<B_2>$ druhý a $<B_3>$ třetí byte vícebytové instrukce, [(D) (E)] značí obsah paměťové buňky, jejíž adresa je obsažena v registrech D a E. PC značí čítač instrukcí. Jelikož sčítání se normálně provádí ve dvojkovém kódu, zkoumá se instrukcí DAA přenos ze čtvrtého a osmého bitu slova mikropočítače a v závislosti na tomto přenosu prováděna korekce tak, aby sčítání bylo ve dvojkově desítkovém kódu.

Chtějme sečíst čísla X = 0002451825104215,

Y = 0000062514232517. Na počátku výpočtu jsou tato čísla uložena v paměti s libovolným výběrem, jak je uvedeno v tab. 5.5. Program ve strojovém kódu pro uvažovaný součet dvou čísel je v tab. 5.6. První instrukce programu je uložena na adrese 1000_{16}, celý program zaujímá 19 bytů.

Je zřejmé, že i tento jednoduchý, poměrně krátký program je ve strojovém kódu značně nepřehledný. Další nesnáz způsobí potřeba vsunutí instrukce. Znamená to přečíslování adres. Pokud bychom tuto instrukci vkládali mezi adresy 1009_{16} a 1010_{16}, je nezbytné též změnit označení adresy, na kterou se provádí skok.

Program v jazyku symbolických adres pro Intel 8080 MCS-80 pro sečtení uvažovaných čísel X a Y je v tab. 5.7.

5.6 Násobení a dělení pomocí posuvu

U mikroprocesoru 8080 podobně jako u jiných mikroprocesorů násobení se může programovat dvojím způsobem: opakovaným sčítáním nebo pomocí posuvu. Předností prvního způsobu je jednoduchost, předností druhého způsobu je vyšší rychlost. Program pro násobení pomocí posunu vychází z ručně psaného násobení. Uveďme si příklad násobení dvou jednobytových čísel:

násobenec A	násobitel B
0 0 1 0 1 0 1 0	. 0 0 1 1 1 1 0 0

0 0 0 0 0 0 0 0	$= B_7 . A . 2^7$
0 0 0 0 0 0 0 0	$= B_6 . A . 2^6$
0 0 1 0 1 0 1 0	$= B_5 . A . 2^5$
0 0 1 0 1 0 1 0	$= B_4 . A . 1$
0 0 1 0 1 0 1 0	$= B_3 . A . 2^3$
0 0 1 0 1 0 1 0	$= B_2 . A . 2^2$
0 0 0 0 0 0 0 0	$= B_1 . A . 2^1$
0 0 0 0 0 0 0 0	$= B_0 . A . 2^0$
0 0 0 1 0 0 1 1 1 0 1 1 0 0 0	

V pravé části násobení jsou uvedeny jednotlivé sčítance násobení, kde B_0 je nejnižší bit a B_7 nejvyšší bit násobitele. Výsledek násobení lze též zapsat ve tvaru

$$B_0 . A . 2^0 + B_1 . A . 2^1 + ... + B_7 . A . 2^7 \qquad (1)$$

Výpočet tohoto vztahu rozložíme na následující kroky, přičemž předpokládáme, že výsledek sestává ze dvou bytů:

a) testuje se bit nejnižšího řádu násobitele, tj. B_0. Je-li tento bit nulový, přejde se na bod b) . Je-li jedničkový, přičte se

násobenec k nejvyššímu bytu výsledku.

b) posune se dvoubytový výsledek doprava o jedno místo

c) opakují se body a) a b) pro jednotlivé další bity násobitele, až se vyčerpá všech 8 bitů násobitele, tj. bity B_1 až B_7.

Při prvním kroku, tj. při prvním použití bodů a) a b) obdržíme $(B_0 . A . 2^8) . 2^{-1}$,
při druhém kroku

$$[(B_0 . A . 2^8) . 2^{-1} + B_1 . A . 2^8] 2^{-1} =$$
$$= B_0 . A . 2^6 + B_1 . A . 2^7$$

a tak dále až při osmém kroku

$$B_0 . A . 2^0 + B_1 . A . 2^1 + B_2 . A . 2^2 + ...+ B_7 . A . 2^7$$

což je vztah totožný s (1). Postup výpočtu předcházejícího příkladu násobení je uveden v tab. 5.8.

Uvažovaný postup násobení dvou jednobytových čísel bez znaménka můžeme uskutečnit pomocí mikroprocesoru 8080 následovně: obecný pracovní registr B vyhradíme pro vyšší byte výsledku a registr C pro nižší byte výsledku. Posun 16 bitového výsledku o jedno místo vpravo se uskuteční dvěma instrukcemi RAR (jednou instrukcí RAR se posouvá obsah registru P, druhou instrukcí RAR obsah registru C). Obecný pracovní registr D obsahuje násobence, registr C násobitele. Program je v tab. 5.9.

Podobný postup lze použít pro dělení šestnáctibitového (dvoubytového) čísla osmibitovým číslem. V tomto případě musíme nahradit sčítání odčítáním a posun (rotaci) o jedno místo doprava posunem o jedno místo doleva. Příslušný program je v tab. 5.10. Registr B je použit pro vyšší byte dělence, registr C pro nižší byte dělence, registr D je použit pro dělitele. Podíl se ukládá do registru C, zbytek do registru B.

5.7 Skok v programu v závislosti na shodě porovnávaných slov

Mějme v paměti mikropočítače uložených N testovacích slov a porovnávejme s nimi vstupní slovo. Nastane-li shoda vstupního slova s některým z N testovacích slov, pak dojde ke skoku na podprogram označený PODP1, nenastává-li shoda vstupního slova ani s jedním z N testovacích slov, uskuteční se skok na podprogram PODP2. Úlohu můžeme zobecnit ještě tím, že požadujeme skok na

podprogram PODP1 i v případě, že nastává shoda vstupního slova s některým z testovacích slov jen v určitých bitech, které se nekryjí s maskou (tab. 5.11); na maskovaných bitech nezáleží.

Program při použití mikroprocesoru 8080 je uveden v tab. 5.12. Jednotlivá testovací slova jsou porovnávána při oběhu smyčkou. Počet testovacích slov N se uloží do registru C; pomocí něhož se zajistí výstup ze smyčky.

Následuje vložení adresy masky 1 testovacího slova 1 do adresovacího registru tvořeného registry H a L a vstup slova ze vstupního zařazení čís. 0 do střadače. Provede se logický součin slova "maska 1" a vstupního slova. Testují se jen ty bity vstupního slova, v jejichž polohách má slovo "maska 1" stavy 1.

Instrukce INX H zvětší obsah adresového registru o jedničku a tím se získá adresa testovacího slova 1. Testovací slovo se pak porovnává s maskovaným vstupním slovem. Jsou-li obě slova shodná, nastaví se příznak nuly a následující instrukce vyvolá skok na instrukci PODP1. Nejsou-li obě slova shodná, obsah registru C se zmenší o jedničku. Touto instrukcí jsou též ovlivňovány stavy klopných obvodů příznaků kromě klopného obvodu přenosu. Nejsou-li vyčerpána všechna testovací slova, program pokračuje instrukcí INX H a instrukcí skoku na návěští VST; následuje zkoumání všech zbylých testovacích slov. Je-li vyčerpáno všech N testovacích slov, nastaví se během instrukce DCR C klopný obvod příznaku nuly do stavu 1 a instrukce JZ PODP2 zajistí skok na instrukci s návěštím PODP2.

5.8 Větvení programu pomocí tabulky větvení a nepřímého adresování

Je-li třeba zkoumat více za sebou jdoucích podmínek větvení programu a v závislosti na jejich splnění nebo nesplnění provést skok na jednotlivé podprogramy nebo pokračovat ve vyhodnocování dalších podmínek, lze postupovat podle obr. 5.3.

Účinnější způsob je použití tabulky větvení, která obsahuje adresy podprogramů, tj. adresy instrukcí, na které má být proveden skok. Jednotlivé podprogramy jsou nepřímo adresovány.

Předpokládejme, že chceme uskutečnit jeden z osmi podprogramů PODP1 až PODP8 v závislosti na poloze jedničky ve střadači podle tab. 5.13. Tabulka větvení nechť je uložena v paměti tab. 5.14. Jelikož uvažujeme délku slova mikropočítače 1 byte, jsou 16 bitové adresy podprogramů rozděleny na dvě části; $\langle PODP1 \rangle_1$, $\langle PODP1 \rangle_2$, $\langle PODP2 \rangle_1$, $\langle PODP2 \rangle_2$ atd. Uvažovaný program pro mikroprocesor 8080 je v tab. 5.15. Příslušný vývojový diagram na obr. 5.4.

5.9 Přenos dat do podprogramů

Podprogram často potřebuje data pro provedení výpočtů. V jednoduchém případě mohou být tato data přenášena do podprogramů pomocí jednoho nebo více obecných pracovních registrů. Jako příklad si uveďme podprogram ZVETS tab. 5.16 , který přejímá adresy paměti pomocí registrů H a L. Tento podprogram zvětšuje 16 bitové číslo o jedničku. Toto číslo je uloženo v sousedních adresách, nižší byte první. Po ukončení podprogramu nastává návrat do původního programu na instrukci následující za instrukcí, která vyvolala podprogram ZVETS. Adresa čísla, které má být zvětšeno, je v registrech H a L.

Mnohdy je účelnější, aby obecné pracovní registry využíval i podprogram. Jedna z možností pro přenos požadovaných dat do podprogramu je použití tabulky, kterou nazýváme seznam parametrů (parametr list). Tento seznam parametrů se uloží do určité části paměti. Data uvedená v seznamu parametrů se předávají do podprogramu pomocí registrů H a L, ve kterých je uložena adresa seznamu parametrů, tj. adresa prvního údaje v tabulce.

Jako příklad si uveďme podprogram SOUC pro sečtení dvou jednobytových čísel uložených na sousedních adresách v operační paměti, výsledek se ukládá na následnou třetí adresu operační paměti tab. 5.17 . Tyto tři následné paměťové buňky představují seznam parametrů. Tento seznam je určen adresou seznamu parametrů obsaženou v registrech H a L. Těchto seznamů parametrů může být více, v uvažovaném programu jsou dva, nazvané SEZN1, SEZN2. Pomocí prvního seznamu parametrů SEZN1 se provádí součet $4_{10} + 2_{10}$, pomocí druhého seznamu parametrů SEZN2 se provádí součet $0A_{16} + 23_{16} = 10_{10} + 35_{10}$.

Při prvním vyvolání podprogramu SOUC se přenese obsah paměťové

buňky s adresou SEZN1 do střadače A a obsah paměťové buňky s adresou SEZN1 + 1 do obecného pracovního registru B. Následuje součet obsahů A a B a výsledek se uloží na adrese SEZN1 + 2. Návrat do podprogramu je proveden na instrukci označenou NAVR1.

Při druhém vyvolání podprogramu je adresován obsahy registrů H a L seznam parametrů SEZN2. Do střadače A se zavede $0A_{16}$ do registru B 23_{16}. Výsledek $2D_{16}$ (= 45_{10}) se uloží na adrese SEZN2 + + 2. Návrat do programu se provede na instrukci s návěštím NAVR2.

5.10 Sečtení libovolného počtu čísel pomocí podprogramu a seznamu parametrů

Sčítání dvou čísel pomocí podprogramu a seznamu parametrů uvedené v předcházejícím příkladu má určité omezení. Podprogram SOUC může provádět součet pouze dvou čísel, která musí být v operační paměti.

Uvažujme příklad, v němž požadujeme sečíst libovolný počet bytů umístěných kdekoliv v paměti, s výsledkem ve střadači. Příslušný podprogram SOUCO je v tab. 5.18. Způsob přenosu dat do podprogramu je znázorněn na obr. 5.5. V tomto programu se předává **podprogramu** pomocí seznamu parametrů seznam adres parametrů, kdežto v předcházejícím příkladu byly v seznamu parametrů přímo obsažena data. Konec seznamu parametrů indikuje adresa, jejíž první byte je FF_{16}, tedy žádný parametr není uložen na adrese $FF00_{16}$ nebo vyšší.

Podprogram SOUCO uchovává součty při jednotlivých oběžích smyčkou v registru C. Adresa prvního parametru se uloží do dvojice registrů D a E. Instrukce CPI zkoumá, jsou-li vyčerpány všechny adresy. V případě vyčerpání adres se provede skok na návěští ZPET a po přesunu součtu z registru C do střadače A se uskuteční návrat do původního programu. Nejsou-li vyčerpány všechny adresy v SEZAD, tj. nejsou-li sečtena všechna čísla, pokračuje podprogram přenesením dalšího parametru do střadače a přičtením obsahu registru C k obsahu střadače A. V dalším kroku se zvětší obsah registrů H a L, čímž se získá adresa dalšího parametru, který má být přičten. Nepodmíněný skok na návěští SMYCK zajistí případné přičtení dalších parametrů.

5.11 Vícebitové logické operace

Logické operace mezi dvěma proměnnými A a B lze názorně vyjádřit pravdivostní tabulkou tab. 5.19 . Jelikož slova dat A i B jsou jednobitová, nazýváme i tyto logické operace jednobitové. Vícebitové logické operace se provádějí s vícebitovými slovy dat tak, že mezi odpovídajícími bity jednotlivých slov A, B se provádějí jednobitové logické operace. Délka slova výsledku logické operace je stejná jako délka slova A a B.

Uvažujme mikroprocesor, jehož délka slova je jeden byte. Jednotlivé bity slova A si označme A_0, A_1, ..., A_7; jednotlivé bity slova B si označme B_0, B_1, ..., B_7 a jednotlivé bity výsledného slova Y si označme Y_0, Y_1, ..., Y_7. Pomocí tohoto označení si můžeme např. logický součin y = A. B rozepsat na operace mezi jednotlivými bity

$$Y_0 = A_0 \cdot B_0$$
$$Y_1 = A_1 \cdot B_1$$
$$\vdots$$
$$Y_7 = A_7 \cdot B_7$$

Jinými slovy vícebitová logická operace sestává z několika jednobitových operací. Jako příklad uvažujme logický součet slov A, B, tj. Y = A + B

$$A = A_7\ A_6\ A_5\ A_4\ A_3\ A_2\ A_1\ A_0 = 1\ 0\ 1\ 1\ 0\ 0\ 1\ 1$$
$$B = B_7\ B_6\ B_5\ B_4\ B_3\ B_2\ B_1\ B_0 = 0\ 0\ 1\ 1\ 1\ 1\ 0\ 1$$
$$Y = Y_7\ Y_6\ Y_5\ X_4\ Y_3\ Y_2\ Y_1\ Y_0 = 1\ 0\ 1\ 1\ 1\ 1\ 1\ 1$$

Vícebitové logické operace se s výhodou používají pro vyjádření dvoustavových veličin např. ventil otevřen - uzavřen . Uvažujme osm dvoustavových veličin, které chceme uložit do paměti s později zpracovat. Při jednoduchém postupu bychom mohli jednotlivé dvoustavové veličiny uložit do osmi osmibitových slov. V každém slovu by zůstalo 7 bitů nevyužitých. Ekonomičtější je uložit všech osm bitů do jednoho slova, které nazýváme stavové slovo. Je-li délka stavového slova osm bitů, nazýváme jej někdy stavový byte.

Provedeme-li přiřazení podle tab. 5.20, pak stavový byte 1 0 0 1 0 1 1 0 značí: ventil M je otevřen, ventil N je uzavřen, tlak v bodě R je nad horní hranicí, teplota v bodě R je pod horní hranicí, koncentrace v bodě S je nad horní hranicí, vypínač T je rozepnut, nádoba Q není plná a čerpadlo U je zapnuto.

Pomocí logických operací můžeme např. zjistit, které veličiny se změnily vzhledem k poslednímu měření, jaké jsou tyto změny atd. Předpokládejme, že současný stavový byte A_I = 1 0 0 1 0 1 1 0 a stavový byte zjištěný během předcházejícího měření je A_{II} = = 1 0 0 1 1 0 1 0. Změny stavů jednotlivých veličin od posledního měření můžeme např. zjistit logickou operací neekvivalence (Exclusive -OR)

$$\begin{array}{rl} & A_{II} = 1\ 0\ 0\ 1\ 1\ 0\ 1\ 0 \\ & \underline{A_I = 1\ 0\ 0\ 1\ 0\ 1\ 1\ 0} \\ A_{III} = A_{II} \oplus A_I & = 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0 \end{array}$$

Slovo A_{III} nám říká, že se změnil stav vypínače T a též se změnila koncentrace v bodě S.

Chtějme nyní zjistit, které veličiny změnily stavy z 1 na 0. Tuto změnu zjistíme logickým součinem $A_{II} \cdot A_{III}$

$$\begin{array}{rl} & A_{II} = 1\ 0\ 0\ 1\ 1\ 0\ 1\ 0 \\ & \underline{A_{III} = 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0} \\ A_{IV} = A_{II} \cdot A_{III} & = 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0 \end{array}$$

Ze slova A_{IV} stanovíme, že během doby, která uplynula od předcházejícího měření do současného měření klesla koncentrace v bodě S ze stavu "nad horní hranicí" na stav "pod horní hranicí".

Chceme-li zjistit, které veličiny se změnily ze stavu 0 do stavu 1, provedeme logický součin $A_{III} \cdot \overline{A_{IV}}$

$$\begin{array}{rl} & A_{III} = 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0 \\ & \underline{\overline{A_{IV}} = 1\ 1\ 1\ 1\ 0\ 1\ 1\ 1} \\ A_V = A_{III} \cdot \overline{A_{IV}} & = 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0 \end{array}$$

Slovo A_V říká, že vypínač T sepnul. Stejný výsledek bychom mohli získat logickým součinem $A_I \cdot A_{III}$

$$\begin{aligned} A_I &= 1\,0\,0\,1\,0\,1\,1\,0 \\ A_{III} &= 0\,0\,0\,0\,1\,1\,0\,0 \\ \hline A_V = A_I \cdot A_{III} &= 0\,0\,0\,0\,0\,1\,0\,0 \end{aligned}$$

Jak jsme si na předcházejících příkladech ukázali, lze logickými operacemi zjistit změny jednotlivých veličin. Z těchto změn můžeme pomocí programu vyvodit takové zásahy do řízeného procesu, že tento proces je řízen v požadovaných mezích.

Uvažujme mikroprocesor 8080, který má 28 logických instrukcí. Jeden z dvou operandů, mezi nimiž se provádí logická operace (součin, součet neekvivalence) je ve střadači, výsledek se ukládá též do střadače.

Osm různých instrukcí pro provedení logického součinu má následující strukturu

1 0	1 0 0	S S S
skupina aritmetických a logických operací	operace logického součinu	3 bitový kód pro označení zdrojového registru

kde SSS je tříbitové označení jednoho z obecných pracovních (zápisníkových) registrů nebo obsah paměťové buňky mikroprocesoru 8080, ve kterém je uložen jeden operand, druhý operand je uložen ve střadači. Přiřazení jednotlivých registrů je v tab. 5.21. Mnemotechnické označení pro logický součin je ANA (z angl. And register with A; logický součin střadače a jednoho z obecných pracovních registrů).

Struktura instrukcí pro logický součet a neekvivalenci je podobná jako uvedená struktura pro logický součin. Pro logický součet se používá mnemotechnické označení ORA (Or register with A; logický součet střadače a jednoho z obecných pracovních registrů), instrukce má tvar

1 0 1 1 0 S S S

operace logického součtu

Pro logickou operaci neekvivalence se používá mnemotechnické označení XRA (Exclusive Or register with A; neekvivalence obsahu střadače a jednoho z obecných pracovních registrů), instrukce má tvar

1 0 1 0 1 S S S

operace neekvivalence

Příklady logických operací jsou v tab. 5.22.

Negace obsahu střadače se provádí instrukcí CMA (Complement A;)která v osmičkové soustavě má tvar 057. Poslední tři z 28 logických instrukcí se týkají případů, kdy jeden z operandů je obsažen přímo v instrukci; tab. 5.23.

Uvnitř mikroprocesoru 8080 logické operace se uskutečňují v dočasném střadači, odkud se výsledek přenese do registru A, tj. do střadače.

5.12 Převod číslic z kódu ASCII do kódu 8 - 4 - 2 - 1 a ukládání dvou číslic v kódu 8 - 4 - 2 - 1 do jednoho bytu

Uvažujme číslice 0 až 9 v kódu ASCII tab. 5.24 . Pro zpracování těchto číslic v mikropočítači není třeba uvažovat všechny bity. Postačuje použít jen čtyři nejnižší bity, čímž obdržíme jednotlivé číslice vyjádřené v kódu 8 - 4 - 2 - 1. Číslice vyjádřené v kódu 8 - 4 - 2 - 1 získáme z číslic vyjádřených v kódu ASCII logickým součinem masky 00001111 a kódu ASCII, např.

$$\underbrace{10110111}_{7_{10}\text{ v kódu ASCII}} \cdot \underbrace{00001111}_{\text{maska}} = \underbrace{00000111}_{7_{10}\text{ v kódu 8 -4 -2- 1}}$$

Program pro mikroprocesor 8080, který provádí uvažovaný převod je

je v tab.5.25. V tomto případě jsou vždy 4 nejvyšší bity slova nevyužity, neboť slovo obsahuje jen jednu číslici. Pokud chceme šetřit kapacitu paměti, lze do osmibitového slova uložit dvě číslice. Program pro 8080 je v tab. 5.26.

6. ANALOGOVĚ ČÍSLICOVÉ A ČÍSLICOVĚ ANALOGOVÉ PŘEVODNÍKY S VAZBOU NA MIKROPOČÍTAČ

Analogově číslicové a číslicově analogové převodníky (AČ, AD a ČA, DA převodníky) nacházejí použití všude tam, kde je třeba analogový signál číslicově zpracovat (AČ převodníky) nebo vytvořit (ČA převodníky). I když se některé analogové veličiny např. úhel mohou měřit přímo v číslicovém tvaru, většina z nich je nejprve převádí vhodným čidlem na elektrickou analogovou veličinu (napětí nebo proud) a pak se tato elektrická analogová veličina převede na číslicový tvar. Poznamenejme, že většina přirozených zdrojů informací má analogový charakter, např. lidský hlas, tlak (plynu, kapaliny), změna objemu či váhy, pohyb, teplota, šedá stupnice televizního obrazu.

Číslicové zpracování analogových signálů má řadu výhod, které jsou umocněny dostupností poměrně laciného číslicového zařízení - mikropočítače. Uveďme si některá použití AČ a ČA převodníků:

- inteligentní měřicí přístroje (výpočet střední hodnoty, výpočet efektivní hodnoty atd.)
- snímání měřených hodnot a jejich zapamatování (periodický záznam měřených veličin ve sledovaném procesu pro pozdější další zpracování)
- vytváření různých napěťových průběhů pomocí programového vybavení (buzení souřadnicového zapisovače, servomotoru)
- přístroje pro zkoušení analogových zařízení, která vyžadují složité, současnými generátory prakticky nerealizovatelné časové průběhy napětí.

Převodová charakteristika ideálního analogově číslicového převodníku je na obr. 6.1. Analogové vstupní napětí je kvantováno do 2^n úrovní, analogově číslicovým převodníkem s n bitovým vstupním slovem. Analogově číslicový převodník má $2^n - 1$ nenulových

rozhodovacích úrovní. Kvantizační krok Q je roven plnému rozsahu převodníku dělenému $2^n - 1$. Kvantizační krok je též roven nejnižšímu bitu (nejnižšímu dvojkovému řádu) výstupního slova a odpovídá rozlišovací schopnosti převodníku. Maximální chyba převodu pro ideální převodník je $\pm Q/2$.

Jednoduchý a velmi často používaný způsob propojení mikropočítače s "analogovým okolím" je znázorněn na obr. 6.2. V tomto případě má mikropočítač jeden analogový vstup a jeden analogový výstup, který může být např. linearizovanou verzí vstupu. Je řada možností, jak převést analogovou veličinu na číslicovou a naopak. V některých případech je mikropočítač plně vytížen jinými úlohami, takže nemůže hrát žádnou roli v obou převodech, pak jen čte a vydává data. Oba převody zajišťují vnější obvody (převodníky) s vhodnými vazebními obvody.

V případech, kdy mikropočítač není plně vytížen řízeným procesem, může se více či méně využít pro dílčí funkce převodníků. V tomto případě nahrazujeme vnější technické vybavení programovým vybavením. Přesněji řečeno, analogově číslicový i obrácený převod vždy znamená určitý rozsah vnějšího technického vybavení a určitý rozsah programového vybavení. O jejich rozsahu je třeba předem rozhodnout.

6.1 Číslicově analogový převod řešený programovým vybavením

Číslicově analogový převodník lze řešit plně pomocí programového vybavení generováním sledu impulsů s faktorem plnění poměr šíře impulsu a prodlevy mezi sousedními impulsy . úměrným číslicovému vstupu. Výstupní analogové napětí se získá nízkofrekvenční propustí, např. obvodem RC. Toto řešení však značně časově zatěžuje mikropočítač a proto při dostupnosti relativně laciných monolitických číslicově analogových převodníků je zpravidla nevhodné.

6.2 Číslicově analogový převod řešený technickým vybavením

Při řešení číslicově analogového převodu je třeba ČA převodník doplnit pamětí dat, obvodem pro dekódování adresy a řídicím obvodem. Má-li monolitický jednočipový mikropočítač vhodné

vstupy - výstupy např. typ 8048 fy Intel , může být převodník připojen přímo k jeho vstupu - výstupu bez dalších obvodů.

Mnohdy číslicově analogový převodník představuje z hlediska adresování jednu paměťovou buňku, což nazýváme jednosběrnicové uspořádání. Vazba paměti dat, obvodu pro dekódování adresy a ČA převodníku na mikroprocesor 8080 je na obr. 6.3. Pro získání nové analogové hodnoty je třeba použít instrukce MOV, která zajistí naplnění paměti dat registru dat novou informací. Je-li použit číslicově analogový převodník s délkou slova 8 bitů, stačí jedna instrukce MOV, při délce slova 10 resp. 12 bitů jsou třeba dvě.

Některé monolitické číslicově analogové převodníky obsahují též obvody pro dekódování adresy, potřebné řídicí obvody a paměť dat. Příkladem může být typ MP 10 firmy Burr-Brown. V jednom pouzdru s 32 vývody jsou dva takto vybavené nezávislé číslicově analogové převodníky. Každý převodník je považován za jednu paměťovou buňku (jednosběrnicové uspořádání). 26 vývodů tohoto integrovaného obvodu se připojuje k vývodům mikroprocesoru 8080, dva vývody jsou pro analogový výstup, dva pro napájení +15, - 15 V, a dva vývody pro adresu, která odpovídá převodníkům. Při použití převodníku MP 10 ve spojení s mikroprocesorem 8080 jedna instrukce SHLD (přímé uložení obsahů registru H a L do paměti) přenáší data z registru H mikroprocesoru do jednoho ČA převodníku a z registru L do druhého ČA převodníku. Nebo instrukce STA (přímé uložení obsahu střadače do paměti) může být použita pro přenos obsahu střadače do jednoho z ČA převodníku.

Doplněním integrovaného obvodu MP 10 napěťovým komparátorem lze tento celek použít jak pro analogový vstup tak pro analogový výstup. Analogově číslicový převod je v tomto případě řešen pomocí programového vybavení použitím metody postupné aproximace.

6.3 Generování složitějších napěťových průběhů pomocí mikropočítače a ČA převodníku

Číslicově analogový převodník, který je na výstupu mikropočítače, často budí další zařízení, např. grafickou zobrazovací jednotku displej , zapisovač, servomotor, ventil atd. Mnohé

průběhy analogového napětí, např. pilový a obdélníkový, lze vytvořit pomocí programu, jehož základem jsou především instrukce zvětšení a zmenšení o jedničku a zpoždění. Takto vytvořená analogová napětí jsou dobře definována.

Pro některé aplikace jsou žádoucí složitější průběhy analogového napětí; tyto složitější průběhy mohou být generovány dvěma způsoby:

1. Výpočtem
   Pro výpočet jednotlivých bodů průběhu jsou sestaveny vhodné podprogramy, nebo další bod se stanovuje výpočtem z poloh předcházejících bodů. Je-li však výpočet jednotlivých bodů složitý, může být generování napěťového průběhu pro některé aplikace nepřijatelně dlouhé. Např. generování trigonometrických funkcí může trvat několik milisekund. V tomto případě může být řešením výpočet průběhu s uložením výsledků do paměti a v další fázi přenos uložených dat do převodníku.
2. Uložením průběhu do paměti
   Hodnoty souřadnic analogového průběhu lze uložit např. do programovatelné nebo reprogramovatelné permanentní paměti mikropočítače. Je-li třeba, je možné programově zajistit, aby se tento složitý analogový průběh periodicky opakoval.

6.4 Princip funkce číslicově analogových převodníků

Číslicově analogové převodníky s napěťovým výstupem jsou nejčastěji řešeny s pomocí váhové nebo příčkové struktury odporové sítě.

Princip číslicově analogového převodníku s napěťovým výstupem a váhovou strukturou odporové sítě si snadno vysvětlíme pomocí obr. 6.4. Vstupní slovo vyjadřuje číslo. Jednotlivé bity vstupního slova ovládají přepínače Př 1, Př 2 a Př 3, které mají v sérii odpory R, 2R a 4R. Proudy procházející přepínači se sčítají na společném odporu představovaném vstupním odporem výstupního zesilovače. Při vhodné volbě poměrů odporů, (které jsou zařazeny v sérii s přepínači), mohou jednotlivé přepínače představovat váhy v přirozeném dvojkovém kódu. V našem případě přepínač Př 1 odpovídá nejnižší váze (nejnižšímu bitu, tj. nejnižšímu dvojkovému řádu

ve vstupním slovu, LSB), přepínač Př 3 odpovídá nejvyšší váze (MSB). Vzhledem k poměrům jednotlivých odporů R, 2R, 4R tento převodník nazýváme převodníkem s váhovou strukturou odporové sítě.

V uvažovaném případě je zapojení operačního zesilovače takové, že na jeho vstupu je přibližně nulové napětí. Každý přepínač připojený na referenční napětí U_{ref} v sérii s příslušným odporem představuje zdroj proudu. Na vstupu zesilovače se sčítají jednotlivé proudy. Celkový proud odporovou sítí je roven zpětněvazebnímu proudu $I_{zpět}$

$$I_{zpět} = \frac{U_{ref}}{R} \left(B_1 + \frac{B_2}{2} + \ldots\ldots + \frac{B_n}{2^n} \right)$$

kde $B_i = 0$ nebo 1 $\quad i = 1, 2, \ldots\ldots n$ podle stavu přepínače.

Příklad převodníku s příčkovou strukturou odporové sítě R-R/2 je na obr. 6.5. Vstupní proud z referenčního zdroje U_{ref} se dělí v každém uzlu a odpovídá dvojkové váze. Odpor, který představuje odporová síť pro výstupní zesilovač, je konstantní. Proto zesilovač nemění šíři přenášeného pásma. Jelikož odpory mají srovnatelnou hodnotu, mohou být zhotoveny stejnou technologií např. hybridní obvod , čímž se docílí snadněji stejná teplotní závislost. Jistou nevýhodou uvedeného zapojení je zvýšený počet odporů. Vzhledem ke struktuře odporové sítě převodníku nazýváme jej převodníkem s příčkovou strukturou odporové sítě R-R/2.

Struktura odporové sítě je uspořádána tak, že příspěvek každého následujícího bitu nalevo od každého uzlu je R. V důsledku toho příspěvek každého následujícího bitu k výstupnímu analogovému napětí se vždy zmenšuje s násobkem 0,5. Jelikož odpor nalevo od uzlu 1 je R, výstupní analogové napětí pak při připojení jen S_1 (nejvyšší bit) na referenční napětí U_{ref} je

$$U_{výst\ S_1} = \frac{U_{ref}}{2 + \frac{R}{R_{zat}}} = \frac{U_{ref} \cdot 2^{-k}}{1 + \frac{R}{2\,R_{zat}}} ; \qquad k = 1$$

kde R_{zat} je zatěžovací odpor. Pomocí principu superpozice výstupní analogové napětí $U_{výst\ A}$ při sepnutí libovolných přepínačů na referenční napětí je

$$U_{výstA} = \frac{2\,U_{ref}\,R_{zat}}{2\,R_{zat} + R} \sum_{k=1}^{n} B_k\,2^{-k}$$

kde B_k = 0 nebo 1 (k = 1, 2, 3, ..., n) v závislosti na poloze jednotlivých přepínačů.

Pro některá použití je výhodné nebo žádoucí (např. pro elektromagnetické vychylování elektronového paprsku v obrazovkovém grafickém displeji), aby ČA převodník měl proudový výstup. Příklad zapojení je na obr. 6.6. Výstupní odpor převodníku je $\frac{2}{3}$ R a není závislý na stavu spínačů, neboť s nimi připojujeme zdroje s vysokým výstupním odporem.

Uvažujeme-li n bitový převodník tohoto typu, pak výstupní napětí je určeno vztahem

$$U_{výstA} = \frac{2}{3}\left(B_1 + \frac{1}{3}\,B_2 + \frac{1}{6}\,B_3 + \frac{1}{12}\,B_4 + \ldots\ldots + \frac{\frac{2}{3}}{2^{n-1}}\,B_n\right)\frac{R\;R_{zat}}{\frac{2}{3}R + R_{zat}}\;I$$

kde B_i = 1 nebo 0; i = 1, 2, 3, 4. B_1 odpovídá nejvyššímu bitu vstupního slova.

6.5 Parametry a přesnost číslicově analogových převodníků

Jeden z důležitých parametrů číslicově analogových převodníků je rozlišovací schopnost, kterou vyjadřujeme počet diskrétních stupňů výstupního analogového napětí. Rozlišovací schopnost je v přímé souvislosti s počtem bitů, z kterých sestává převáděné slovo. Osmibitový číslicově analogový převodník má $2^8 = 256$ stupňů, což odpovídá rozlišovací schopnosti 100/256 ≐ 0,4 %. Použijeme-li však přirozeného dvojkově desítkového kódu, pak při zachování celkového počtu osmi bitů je rozlišovací schopnost převodníku 1 %, neboť se dvěma čtveřicemi bitů můžeme vyjádřit dva desítkové řády.

Zvyšujme postupně výstupní analogové napětí ideálního číslicově analogového převodníku po jednotlivých diskrétních stupních.

Výstupní napětí převodníku se mění skokem z jedné napěťové úrovně na druhou. Tato změna výstupního napětí odpovídá nejnižšímu bitu vstupního slova. Chyba způsobená diskrétními úrovněmi výstupního napětí (kvantizační chyba) může dosahovat maximálně ± 1/2 hodnoty odpovídající nejnižšímu bitu vstupního slova.

Dalším důležitým parametrem číslicově analogového převodníku je přesnost analogového výstupního napětí nebo proudu. Příklad ideální převodní charakteristiky tříbitového číslicově analogového převodníku je na obr. 6.7. Tento převodník má osm diskrétních kódovaných vstupních úrovní, z kterých získáme osm úrovní normalizovaného výstupního napětí, jehož rozsah je 0 až 7/8.

V praxi se skutečná převodová charakteristika může lišit od ideální v důsledku různých vlivů. Vliv napěťového posunu je na obr. 6.8. Chyba zisku (způsobená např. změnou zisku zesilovače nebo změnou referenčního napětí) je znázorněna na obr. 6.9. Dalším zdrojem nepřesnosti číslicově analogového převodníku může být nelinearita (obr. 6.10).

Celková přesnost převaděče je podstatně závislá na stabilitě referenčního napětí. Nestabilita napětí referenčního zdroje ovlivňuje přesnost, nemá však vliv na rozlišovací schopnost a na linearitu.

Uveďme si další důležité parametry číslicově analogového převodníku. Maximální rychlost převodu je určena počtem vstupních slov číslicově analogového převodníku, které mohou být tímto převodníkem převedeny na analogové napětí nebo proud za jednotku času. Někdy se též uvádí doba převodu, což je reciproká hodnota rychlosti převodu; je to doba mezi převedením vstupního slova na převodník a okamžikem dosažení ustáleného výstupního napětí (nebo proudu).

Rozsah analogového výstupního napětí je maximální napěťový rozkmit převodníku buď jedné nebo obou polarit. Výstupní impedance převodníku je impedance, která se jeví na výstupních svorkách ve směru do převodníku. Kvantum převodníku je nejmenší přírůstek výstupního napětí převodníku, který můžeme rozlišit dvěma vstupními slovy. Je rovno výstupnímu analogovému napětí, které odpovídá nejnižšímu bitu ve vstupním slovu.

Teplotní koeficient charakterizuje změnu výstupního analogového napětí způsobenou změnou teploty. Vyjadřujeme jej obvykle jako procentuální změnu rozsahu analogového výstupního napětí převaděče v důsledku změny teploty o 1 ^{o}C.

Výstupní napětí číslicově analogového převodníku je schodovité, jeho hodnoty mohou nabývat jen diskrétních hodnot. Tyto diskrétní hodnoty odpovídají jednotlivým úrovňovým stupňům výstupního napětí. Přechod výstupního analogového napětí převaděče z jednoho úrovňového stupně na jiný úrovňový stupeň může být doprovázen přechodovým jevem. Neuvažujeme-li případné překmity zesilovače, přechodový jev je způsoben konečnou dobou sepnutí, resp. rozepnutí polovodičových spínačů.

Obvykle je doba rozepnutí bipolárního tranzistoru delší, než doba sepnutí. V převodníku to může znamenat větší zpoždění při přechodu ze stavu 0 do stavu 1 než opačnému přechodu v jednotlivých bitech vstupního slova. Je-li použit přirozený dvojkový kód, pak při změně vstupního slova z 0111 na 1000 výstupní analogové napětí při přechodovém jevu může dosáhnout na krátký okamžik nulové hodnoty. Tento nežádoucí přechodový jev můžeme potlačit filtrací, což ovšem má za následek zpomalení rychlosti odezvy. Jiným řešením může být zařazení zvláštního obvodu na výstup převodníku, který si zachovává předcházející úroveň do doby ustálení nové úrovně.

6.6 Analogově číslicové převodníky

Převod spojitého analogového signálu na číslicový tvar se provádí ve dvou krocích. Analogový signál se nejprve periodicky vzorkuje, tj. získává se periodický sled úzkých impulsů, jejichž amplitudy odpovídají analogovému signálu v příslušných okamžicích. V druhém kroku jsou amplitudy jednotlivých impulsů převáděny tzv. kvantováním na číslicový tvar (obr. 6.11).

Převod můžeme provést teoreticky s libovolnou přesností. K tomu je třeba, aby:

a) vzorkování analogového signálu bylo prováděno alespoň s dvojnásobným opakovacím kmitočtem než je nejvyšší harmonická složka analogového napětí

b) vzorkovací impulsy byly dostatečně úzké
c) kvantování vzorkovacích impulsů bylo dostatečně "jemné", tj., aby číslo vyjadřující amplitudu mělo dostatečný počet řádů.

Uvedené procesy mají své technické možnosti. Rychlost změny analogového signálu má své meze v rychlosti a v rozlišovací schopnosti kvantovacího obvodu analogově číslicového převodníku.

Funkce analogově číslicového převodníku spočívá v transformaci vstupního analogového napětí U_{vstup} děleného konstantním referenčním napětím U_{ref} na číslicový výstupní signál X

$$X = \left[\frac{U_{vstup}}{U_{ref}} \right]$$

V tomto vztahu X je nejbližší aproximace uvedeného podílu;
$X = B_1 2^{-1} + B_2 2^{-2} + \ldots\ldots\ldots\ldots\ldots + B_n 2^{-n}$
kde $B_i = 1$ nebo 0; i = 1, 2, 3, ...n.
Číslo X sestávající z n bitů nazýváme výstupní slovo analogově číslicového převodníku. Napětí U_{ref} je voleno tak, že X je vždy menší než 1.

6.7 Analogově číslicový převod řešený technickým vybavením

Na světovém trhu je řada monolitických a hybridních analogově číslicových převodníků, které se liší především rychlostí, přesností a cenou. Základní požadavky na tyto převodníky z hlediska jejich použití v systémech s mikropočítači jsou: paralelní výstup jednotlivých bitů výstupního slova a vhodná indikace platnosti dat. Je výhodné, je-li výstup AČ převodníku vyjádřen v druhém doplňku, neboť tento tvar používají všechny mikroprocesory.

Typický analogově číslicový převodník vhodný pro použití v systému s mikroprocesorem má paralelní výstup 8, 10 nebo 12 bitů. Dále poskytuje signály "obsazen", "konec převodu", "data platná" a vyžaduje signál "počátek převodu". Podobně jako u číslicově analogových převodníků je třeba obvod pro dekódování adresy. Pokud součástí převodníku není třístavový výstupní obvod, je třeba jej zapojit vně.

Na obr. 6.12 je znázorněno připojení analogově číslicového

převodníku k mikroprocesoru 8080. Převodník má určitou adresu, která je detekována obvodem pro dekódování adresy. Odpovídá-li adresa na adresovací sběrnici mikroprocesoru 8080 adrese převodníku, má výstup dekodéru adresy logickou hodnotu 0. Má-li současně výstup OUT obvodu 8228 logickou hodnotu 1 (tj. $\overline{OUT} = 0$), spustí se monostabilní klopný obvod, který vyvolá počátek analogově číslicového převodu.

Po dokončení převodu může mikroprocesor číst výstup AČ převodníku za předpokladu, že tento převodník je současně adresován. Před čtením výstupu AČ převodníku musí se ještě mikroprocesor přesvědčit, že převod je již ukončen. Uvažujme přenos dat z výstupu AČ převodníku pomocí přerušení. Podle uvedeného obrázku signál přerušení vzniká působením signálu "konec převodu" (INTA = INTERRUPT ACKNOWLEDGE, potvrzení požadavku na přerušení) a signálu z dekodéru adresy. Signálem přerušení se vyvolá krátký podprogram, který uloží obsah střadače mikroprocesoru, přečte výstup AČ převodníku a vynuluje přerušení.

6.8 Rozdělení analogově číslicových převodníků

Je mnoho typů analogově číslicových převodníků; každý z těchto typů má své výhody a nevýhody. Pro určitou aplikaci je třeba vybrat určitý vhodný typ.

Analogově číslicové převodníky můžeme např. rozdělit na:

a) Analogově číslicové převodníky bez zpětné vazby, které bezprostředně porovnávají vstupní analogové napětí a referenční napětí. Výsledkem porovnání je výstupní slovo analogově číslicového převodníku.

b) Analogově číslicové převodníky se zpětnou vazbou, které porovnávají v porovnávacím obvodu vstupní analogové napětí s analogovým napětím odvozeným z postupně generovaného výstupního slova. Převod je ukončen v okamžiku rovnosti obou porovnávaných napětí.

Analogově číslicové převodníky můžeme též rozdělit na synchronní a asynchronní. U synchronních převodníků probíhá převod vstupního analogového napětí v určitém počtu kroků, které se uskutečňují synchronně s taktovacími impulsy. U asynchronních

analogově číslicových převodníků převod se též může uskutečnit v několika krocích, avšak doba trvání jednotlivých kroků závisí výhradně na časové odezvě dílčích obvodů převodníku resp. jejich zpoždění.

Jiné rozdělení analogově číslicových převodníků je na přímé a nepřímé. Přímé analogově číslicové převodníky převádějí vstupní analogové napětí přímo na výstupní slovo. Nepřímé analogově číslicové převodníky převádějí nejprve určitým obvodem vstupní analogové napětí na jinou analogovou veličinu (např. na dobu trvání impulsu nebo proměnný opakovací kmitočet sledu impulsů) a dalším obvodem takto získanou analogovu veličinu převádějí na číslicový tvar (např. pomocí časového nebo kmitočtového standardu).

6.9 Nejčastěji používané principy analogově číslicových převodníků

Paralelní analogově číslicový převodník

Nejrychlejším a současně principiálně nejjednodušším typem analogově číslicového převodníku je paralelní typ. Čtyřbitový převodník tohoto typu je na obr. 6.13. Analogové napětí je přiváděno současně na všech patnáct napěťových komparátorů (prahových detektorů), neboť je 15 možných úrovní pro uvažovaný čtyřbitový převodník. Na výstupu jednotlivých komparátorů je logická hodnota (stav)1, pokud $U_{vstup} \geq U_{refi}$, i = 1, 2, 3 ....15. V opačném případě je na výstupu napěťového komparátoru logická hodnota (stav) 0.

Převaděč kódu, představovaný kombinačním logickým obvodem, převede výstupy z napěťových komparátorů do dvojkového nebo jiného zvoleného kódu.

Doba převodu paralelního převodníku je určena přenosovým zpožděním, resp. dobou ustálení napěťových komparátorů a přenosovým zpožděním v kombinačním logickém obvodu (může být několik desítek ns i méně). Pro výstupní slovo sestávající z n bitů je třeba $2^n - 1$ napěťových komparátorů. Pro n větší než 3 až 4 cena paralelního převodníku bývá větší než jiných typů. Proto se používá zřídka.

Analogově číslicový převodník s postupnou aproximací

Funkce analogově číslicového převodníku, který pracuje na principu postupné aproximace, je zřejmá z obr. 6.14. Výstupní slovo uvažovaného převodníku sestává ze čtyř bitů, tj. při použití přirozeného dvojkového kódu rozsah vstupního napětí je rozdělen na 15 kvantizačních úrovní.

Analogově číslicový převod se uskutečňuje v našem případě ve 4 krocích. Pro jednoduchost předpokládáme, že rozsah vstupního analogového napětí je 0 až 15 V. V prvním kroku převodník rozhoduje, zda vstupní analogové napětí je větší nebo menší než 8 V (1/2 plného rozsahu). Je-li vstupní analogové napětí U_{vstup} větší než 8 V, první (nejvyšší) bit výstupního slova převodníku je 1, v opačném případě 0.

Během druhého kroku za předpokladu $U_{vstup} > 8$ V je opět rozdělena horní polovina rozsahu. Převodník rozhodne, zda vstupní analogové napětí je menší nebo větší než 12 V a vytvoří další bit výstupního slova. Po provedení všech čtyř kroků získáme na výstupu převodníku všechny 4 bity výstupního slova. Jako příklad je v obrázku znázorněn převod vstupního analogového napětí 8,4 V. V obecném případě, kdy výstupní slovo sestává z n bitů, kvantujeme vstupní analogové napětí v n krocích.

Skupinové schéma analogově číslicového převodníku s postupnou aproximací je na obr. 6.15. Jeho součástí je číslicově analogový převodník, který zpětně převádí výstupní slovo $B_8\ B_4\ B_2\ B_1$ analogově číslicového převodníku na srovnávací napětí U_{srov}. Toto srovnávací napětí U_{srov}, které se mění v každém kroku, je porovnáváno v napěťovém komparátoru se vstupním analogovým napětím U_{vstup}. Funkci řídicího obvodu i registru může provádět mikropočítač.

Analogově číslicový převodník s dvojnásobným pilovým průběhem a integračním obvodem

Příklad tohoto převodníku je na obr. 6.16, časový průběh na výstupu integračního obvodu je na obr. 6.17. Na počátku převodu ($t = 0$, resp. $t = k \cdot T_2$, kde $k = 0,1,2,3,\ldots\ldots$) je vynulován čítač a sepnut spínač představovaný tranzistorem T_1. Výstupní

napětí integračního obvodu $U_{int} = -U_p$ (v ideálním případě $U_{int} = 0$) vyjadřuje nepřesnost napěťového komparátoru.

Analogově číslicový převod se uskutečňuje ve dvou fázích. V první fázi činnosti čas $0 < t < T_1$ je vodivý tranzistor T_1, vstupní analogové napětí integruje integrační obvod, který sestává z odporu R, kapacity C a zesilovače. Během této fáze činnosti výstupní napětí z napěťového komparátoru otevře součinové hradlo (součinový logický člen) a čítač čítá impulsy ze zdroje hodinových impulsů. Po naplnění čítače následující hodinový impuls vyvolá "přetečení" čítače. Impuls "přetečení" změní stav klopného obvodu, tranzistor T_1 se uzavře a T_2 otevře (čas $t = T_1$).

V čase T_1 je výstupní napětí z integračního obvodu

$$U_{int} = \frac{1}{RC} \int_0^{T_1} U_{vstup}\, dt - U_p = \frac{1}{RC} \cdot \overline{U_{vstup}} \cdot T_1 - U_p ,$$

kde $\overline{U_{vstup}}$ je střední hodnota vstupního analogového napětí v intervalu $0 < t < T_1$. Jelikož T_1 je předem stanoveno přetečení čítače, U_{int} je úměrné U_{vstup}.

Během druhé fáze činnosti analogově číslicového převodníku, tj. během doby $T_1 < t < T_2$ integrační obvod integruje referenční napětí $-U_{ref}$. Jeho výstupní napětí je

$$U_{int} = -U_p + \frac{1}{RC} \overline{U_{vstup}}\, T_1 - \frac{1}{RC} \int_{T_1}^{t} U_{ref}\, dt$$

kde $-U_{ref}$ je konstantní referenční napětí. Integrování referenčního napětí pokračuje do okamžiku, kdy napěťový komparátor změní stav. V tomto okmažiku je výstupní napětí integračního obvodu $-U_p$ a podle předcházejícího vztahu

$$U_{int} = -U_p + \frac{1}{RC} \overline{U_{vstup}}\, T_1 - \frac{1}{RC} U_{ref} (T_2 - T_1)$$

Úpravou tohoto vztahu obdržíme

$$U_{vstup} = U_{ref} \cdot \frac{T_2 - T_1}{T_1}$$

V době $t = T_1$ došlo k přetečení čítače a tím současně k jeho vynulování. Nechť v době $t = T_2$ čítač napočítá N impulsů.
Pak doba

$$T_2 - T_1 = \frac{N}{f_{opak}}$$

kde f_{opak} je opakovací kmitočet hodinových impulsů. Konstantní doba T_1 je určena vztahem

$$T_1 = \frac{2^n}{f_{opak}}$$

Střední hodnotu vstupního analogového napětí U_{vstup} můžeme vyjádřit vztahem

$$\overline{U_{vstup}} = U_{ref} \frac{T_2 - T_1}{T_1} = U_{ref} \frac{N}{f_{opak}} \cdot \frac{f_{opak}}{2^n} = U_{ref} \frac{N}{2^n}$$

Vstupní napětí $\overline{U_{vstup}}$ je přímo úměrné počtu napočítaných impulsů N během doby $T_2 - T_1$. Předcházející vztah nám ukazuje, že toto vstupní napětí $\overline{U_{vstup}}$ není závislé na U_p (vylučujeme teplotní a napěťovou nestabilitu napěťového komparátoru), dále na f_{opak} (zdroj hodinových impulsů nemusí být dlouhodobě stabilní) a není také závislé na integrační konstantě 1/RC. Další výhodou uvažovaného analogově číslicového převodníku je značná necitlivost k šumu ve vstupním napětí, neboť provádíme integraci vstupního napětí po dobu, která je nezávislá na vstupním napětí.

Zdrojem nepřesností uvažovaného převodníku je nedokonalost spínacích vlastností tranzistorů T_1 a T_2, nepřesnost zdroje referenčního napětí U_{ref} a zpoždění tranzistorů T_1 a T_2. Dalším zdrojem nepřesností je nelinearita integračního obvodu. Přesnost převodníku neovlivňuje skutečné prahové napětí komparátoru. Rovněž zpoždění komparátoru neovlivňuje přesnost, nebo se jeho vliv ruší při integraci U_{vstup} a U_{ref}.

Uvažovaný analogově číslicový převodník s dvojnásobným pilovým průběhem a integračním obvodem můžeme charakterizovat poměrně malou rychlostí převodu, značnou dosažitelnou přesností a obvodovou jednoduchostí bez větších nároků na přesnost mnohých prvků.

Poslední vlastnost vytváří nezbytné předpoklady pro integraci.

Analogově číslicový převodník s proměnným kmitočtem

Jádro tohoto převodníku tvoří napěťově kmitočtový převodník, jehož výstupem je sled stejných impulsů s opakovacím kmitočtem úměrným vstupnímu analogovému napětí. Pro úplný převod vstupního analogového napětí na číslicový tvar je nezbytné za napěťově kmitočtový převodník zařadit čítač impulsů.

Pro napěťově kmitočtový převodník lze použít různé obvody, např. napěťově závislý kondenzátor (varikap) zapojený v laděném obvodu nebo tranzistor řízený polem jako proměnný odpor v oscilátoru RC. Jiné řešení napěťově kmitočtového převodníku je na obr. 6.18. Předpokládáme-li U_{vstup} = -1 V, nabíjí se kondenzátor C_1 přes vstupní odpor R s počáteční rychlostí -1 V/s. Klesne-li napětí U_1 na kondenzátoru na -1 mV, napěťový komparátor spustí monostabilní klopný obvod, na jehož výstupu je nyní jediný impuls o době trvání asi 1 /us. Současně se vynuluje (vybije) pomocí nulovacího obvodu kondenzátor C_1 na 0 V. Protože nabití kondenzátoru na -1 mV při vstupním napětí - 1 V trvá 1 ms, je opakovací kmitočet výstupních impulsů 1 kHz.

Je-li U_{vstup} = -10 V, opakovací kmitočet výstupních impulsů f_{opak} = 10 kHz. Pro U_{vstup} = -10 mV je f_{opak} = 10 Hz, viz obr. 6.19.

6.10 Řešení analogově číslicového převodu programovým vybavením

Není-li mikroprocesor vytížen řízeným procesorem a nevyžaduje-li se značná rychlost převodu, je výhodné z hlediska nákladů řešit analogově číslicový převod programovým vybavením. Nejčastěji se používá metoda postupné aproximace, neboť je relativně značně rychlá a současně vyžaduje minimální technické vybavení vně mikroprocesoru.

Algoritmus postupné aproximace je uveden na obr. 6.20. Potřebné minimální technické vybavení je na obr. 6.21. U tohoto převodníku se nejprve nastaví nejvyšší bit výstupního slova převodníku rovný 1, ostatní bity rovny 0. Toto slovo se přivede na číslicově analogový převodník, jehož výstup je porovnáván se vzorkem vstup-

ního analogového napětí. Je-li výstup ČA převodníku vyšší než vstupní analogové napětí, nejvyšší bit slova se vynuluje, v opačném případě se ponechá (logická hodnota = 1). Tento proces se postupně opakuje pro další bity s nižší váhou.

Tento obecný postup lze použít u libovolného procesoru, avšak způsob provedení jednotlivých kroků závisí na konkrétním použitém technickém vybavení. Na obr. 6.20 je uvažován mikroprocesor (resp. mikropočítač) 8048, příslušný program je v tab. 6.1. Program využívá čtyři registry mikroprocesoru 8048: střadač A a registry R5, R6, R7. Dílčí výsledky jsou uchovávány v registru R6. Registr R5 se používá jako ukazatel bitu výstupního slova převodníku, který má být v dalším kroku zpracován. Registr R7 se používá jako čítač oběhů smyčkou.

Program začíná návěštím CNVT vložením hodnoty 08_{16} do registru R7 pro volbu délky slova AČ převodu 8 bitů. Následuje vynulování střadače a vynulování registrů R5 a R6 instrukcemi MOV R5, A; MOV R6, A. Jelikož mikroprocesor nemá instrukci pro nastavení klopného obvodu příznaku přenosu, je třeba použít postupně dvě instrukce CLR C (vynulování klopného obvodu příznaku přenosu) a CPL C (negace obsahu klopného obvodu příznaku přenosu). Instrukce ANL provádí logický součin dat na vstupu - výstupu P2 s okamžitým operandem 11111110 ($= FE_{16}$). Jejím účelem je vynulovat nejnižší bit na vstupu - výstupu P2.

Výstup z P2 tvoří řídicí vstup paměťového a vzorkovacího obvodu. Jinými slovy instrukce ANL způsobí, že analogové napětí vyskytující se v určitém okamžiku na vstupu je zachováno po dobu celého procesu analogově číslicového převodu. Tento krok je v celkovém algoritmu velmi důležitý, neboť případná změna porovnávaného napětí během procesu převodu by mohla způsobit chybný výsledek.

Prvních sedm instrukcí tvoří počáteční část programu. Instrukce ve smyčce vytvoří při každém oběhu jeden bit výstupního slova převodu. Instrukcí MOV A, R5 se obsah registru přenese do střadače. Instrukcí RRC A se provede posun obsahu střadače o jedno místo vpravo přes klopný obvod příznaku přenosu. Při této instrukci se naplňuje klopný obvod příznaku přenosu nejnižším bitem střadače a obsah klopného obvodu příznaku přenosu je přenesen do nejvyššího bitu střadače. Při prvním oběhu smyčkou se přesune obsah klop-

ného obvodu příznaku přenosu, který byl nastaven na logickou hodnotu 1 do nejvyššího bitu střadače, tj. jeho obsah je 10000000. Další oběhy smyčkou postupně posouvají jedničku ve střadači vždy o jedno místo vpravo; tato jednička ukazuje, který bit výstupního slova má být stanoven. Instrukce MOV R5,A způsobí, že do registru R5 je uložen nový stav ukazatele zpracovávaného bitu. Instrukce ORL A,R6 provádí skutečné nastavení bitu ve vstupním slově, tím, že logicky sečte obsah střadače a obsah registru R6 a výsledek uloží do střadače. Střadač A však obsahuje samé nuly kromě jedné jedničky v řádu, který se právě stanovuje. Výsledkem této operace je, že v řešeném řádu se nastaví jednička, v nižších řádech zůstanou zachovány nuly a ve vyšších řádech zůstane zachován původní obsah. V tomto bodě střadač A obsahuje novou odhadnutou hodnotu AČ převodu a tato hodnota je přivedena na číslicově analogový převodník připojený na vstup - výstup P1 instrukcí OUTL P1, A. Instrukce JTØ způsobí podmíněný skok na návěští TOHI v případě, že výstup z napětového komparátoru má logickou hodnotu 1, což indikuje, že výstup z číslicově analogového převodníku je vyšší než vstupní analogové napětí. V opačném případě se provede instrukce MOV R6,A, která nahradí předcházející obsah registru R6 současným obsahem střadače. Při skoku se tento krok vypustí a původní obsah registru R6 zůstane zachován. Poslední instrukce DJNZ R7, LOOP způsobí zmenšení v obsahu registru R7 o jedničku a porovnání výsledku s nulou. Není-li výsledek nulový, skočí program na návěští LOOP. Výsledek je nulový až v okamžiku, kdy je stanoveno všech osm bitů výstupního slova AČ převodníku. V tomto případě se výsledek uloží v registru R6 a program vystoupí ze smyčky.

Čas potřebný k úplnému převodu je podstatně delší než u převodníku s postupnou aproximací, který je řešen jako samostatný obvod. V mnohých případech však dostačuje rychlost dosažená pomocí programového vybavení. Technické vybavení je v tomto případě velmi jednoduché: sestává z mikropočítače 8048, číslicově analogového převodníku, vzorkovacího a paměťového obvodu a napěťového komparátoru. V případě použití mikroprocesoru 8080 je třeba ještě před číslicově analogový převodník zařadit vhodný vazební obvod.

6.10.1 Nepodmíněně jednoslovný přenos dat z analogově číslicového převodníku

Nepodmíněný jednoslovný přenos, který též nazýváme jednoslovní synchronní přenos, se uskutečňuje s přídavným zařízením, jehož časování je známo. Při tomto přenosu musí být přídavné zařízení (v našem případě AČ převodník) připraveno pro přenos.

Uvažujme 8 bitový AČ převodník označený jako přídavné zařízení 3. Tento převodník nechť je připojen na osmibitovou datovou sběrnici mikropočítače. Předpokládejme dále, že data získaná z převodníku zpracovává mikropočítač a pak je předává na přídavné zařízení 4, které může být představováno ČA převodníkem. V jazyku symbolických adres můžeme psát:

```
... IN 3, ... OUT 4, ...
```

Instrukce IN 3 obsahuje příkaz pro přenesení výstupu analogově číslicového převodníku (přídavné zařízení 3) po datové sběrnici do střadače mikroprocesoru bez ohledu na stav převodníku; předpokládá se, že AČ převodník je připraven k přenosu. Podobně pomocí instrukce OUT 4 se přenáší obsah střadače přes datovou sběrnici na ČA převodník (zařízení 4).

Obvod, který může uskutečnit uvažovaný vstup i výstup je na obr. 6.22. Při instrukci "vstup" se přivede adresa přídavného zařízení 0011 na dekodér 1 ze 16. Aktivační impuls přivedený na dekodér 1 ze 16 aktivuje adresovací vodič přídavného zařízení 3. Následuje přivedení vstupního strobovacího impulsu na součinový logický člen, který aktivuje třístavové logické členy, přes něž se uskutečňuje přenos z AČ převodníku do střadače. Příslušný časový diagram je na obr. 6.23.

Pomocí instrukce OUT 4 se přivede adresa přídavného zařízení 0100 na dekodér 1 ze 16. Aktivační impuls aktivuje adresovací vodič přídavného zařízení 4, kterým se výstupy třístavových logických členů uvedou do stavu nízké impedance. Tím je umožněn přenos ze střadače mikroprocesoru přes datovou sběrnici a třístavové logické členy do vyrovnávací paměti tvořené klopnými obvody typu D. Zápis do tohoto klopného obvodu je podmínen výstupním strobovacím impulsem.

V uvažovaném příkladu jsme předpokládali, že AČ převodník je připraven pro přenos dat v okamžiku, kdy je dán příkaz pro vstup. To však vyžaduje, aby AČ převodník byl nejprve aktivován s časovým předstihem, který je alespoň nepatrně větší, než doba převodu převodníku. Tento předstih lze řešit pomocí dalšího výstupu mikroprocesoru, který aktivuje AČ převodník.

Jako příklad nepodmíněného jednoslovního přenosu si uveďme systém pro sledování osmi analogového napětí, obr. 6.24. Systém sestává z mikropočítače jehož základem je mikroprocesor 8080 , z osmikanálového analogového multiplexoru a osmibitového analogově číslicového převodníku. Mikroprocesor vysílá příkaz pro volbu jednoho z osmi analogových kanálů prostřednictvím výstupu 3 mikropočítače (vodiče O_0, O_1, O_2 resp. A_0, A_1, A_2) . Výstup 3 pomocí vodiče O_4 též vyšle impuls "začátek převodu", kterým se započne analogově číslicový převod. Na počátku převodu je vodič "konec převodu" nastaven do stavu 1. Po dokončení převodu je vodič "konec převodu" nastaven do stavu 0. Mikropočítač trvale čte vstup 2 až do okamžiku, kdy je "konec převodu" ve stavu 0. Následuje přenos výstupu analogově číslicového převodníku přes vstup 3 mikropočítače do střadače a následně do paměti. Postupový diagram pro sestavu programu je na obr. 6.25, příslušný program v tab. 6.2.

V uvažovaném systému vstupy a výstupy mikropočítače invertují data, která jimi procházejí. Tato skutečnost je v programu respektována. Časový diagram analogově číslicového převodníku je uveden na obr. 6.26. Převodník je nulován při přechodu ze stavu 1 do stavu 0 a převod se začíná při přechodu ze stavu 0 do stavu 1 na vodiči "počátek převodu". Konec převodu je signalizován stavem 0 na vodiči "konec převodu".

6.10.2 Vstup dat z analogově číslicového převodníku do mikropočítače podmíněným jednoslovným přenosem

Podmíněný jednoslovný přenos, který někdy nazýváme asynchronní jednoslovný přenos, se zpravidla uskutečňuje v následujících krocích:

1. Vyhodnocení stavu přídavného zařízení, tj. AČ převodníku

2. Aktivace přídavného zařízení v případě signálu "připraven"

3. Přenos dat

4. Uvedení přídavného zařízení do stavu klidu

V prvním kroku se provádí instrukce, kterou se zkoumá stav zvoleného převodníku. Je-li obsazen, je jeho stav pomocí smyčky periodicky zkoumán. Je-li převodník připraven, použije se instrukce vstup - výstup pro vstup dat (3. krok). Po dokončení přenosu je AČ převodník uveden do stavu klidu. Příslušný postupový diagram je na obr. 6.27.

Předpokládejme, že převodník je připojen jako přídavné zařízení 1 (vstup - výstup 1) se stavovým klopným obvodem připojeným jako přídavné zařízení 2 k mikroprocesoru 8080. Stavový klopný obvod AČ převodníku komunikuje na 4 bitovém vodiči osmibitové datové sběrnici mikroprocesoru 8080. Program v jazyku symbolických adres je následující:

```
      :
      .
TEST: IN 2          ; obsah vstupu 2 do střadače
      ANI 10H       ; maska pro 4. bit
      JZ TEST       ; je-li obsazen, skok na TEST
      IN 1          ; přenos dat do střadače
      .
      :
```

V tomto programu instrukce IN 2 přenáší obsah stavového klopného obvodu do střadače. Je-li převodník ve stavu připraven, je 4. bit na datové sběrnici ve stavu 1. Instrukce ANI 10H maskuje všechny bity kromě čtvrtého a v případě, že převodník připraven není, nastavuje tento bit klopný obvod příznaku nuly mikroprocesoru 8080. Instrukce JZ TEST vytváří smyčku až do doby připravenosti zařízení. Pomocí následující instrukce IN 1 vstupují data z převodníku do střadače.

Obvod, který provádí podmíněný přenos podle předcházejícího programu je na obr. 6.28. Instrukce IN 2 je určena pro zkoumání stavu přídavného zařízení AČ převodníku; jeho adresa se přivede na dekodér 1 z n. Aktivační impuls přivedený na tento dekodér aktivuje adresovací vodič přídavného zařízení 2, což je stavový klopný obvod. Strobovací vstupní impuls umožní přenos stavu stavo-

vého klopného obvodu přes 4. bitový vodič datové sběrnice do střadače. Je-li přídavné zařízení představované AČ převodníkem obsazeno, tj. je-li Q = 1, test se opakuje. Je-li převodník připraven (Q = 0), provede se instrukce IN 1. Adresu přídavného zařízení převezme dekodér 1 z n a aktivuje adresovací vodič přídavného zařízení 1. Vstupní strobovací impuls umožní přenos dat z převodníku přes datovou sběrnici do střadače. Příslušný časový diagram je na obr. 6. 29.

V uvažovaném případě se předpokládá, že převod AČ převodníku je již ukončen, když započne sled vstupní sekvence.
Je tudíž třeba před započetím tohoto přenosu vyslat impuls "počátek převodu" do AČ převodníku. Tento impuls může být generován externě nebo mikropočítačem. Je-li generován mikropočítačem, je nezbytná výstupní instrukce, která vyvolá impuls "počátek převodu" prostřednictvím k tomu určenému výstupu z mikropočítače, podobně jako v případě nepodmíněného jednoslovního přenosu.

6.11 Automatické nastavení rozsahu

V některých případech je výhodné nebo žádoucí doplnit systém analogově číslicového a číslicově analogového převodníku jak technickým, tak programovým vybavením, které umožní další funkce, např. automatické nastavení rozsahu, automatickou korekci chyby (driftu) a detekci zvoleného minimálního nebo maximálního napětí.

Obvod na obr. 6.30 umožňuje automatické nastavení rozsahu analogově číslicového převodníku. Zesilovač s programovatelným ziskem, např. řešený pomocí operačního zesilovače s přepínatelnými zpětnovazebními odpory zesiluje vstupní analogové napětí. Spínače, zpravidla tranzistory řízené polem, ovládá mikroprocesor. Je-li základní rozsah vstupního napětí např. 10 V, pak analogové vstupní napětí pod 40 mV při osmibitovém slovu vyvolá nulový výstup. Mikroprocesor může být naprogramován tak, aby zkoumal výstup; je-li nulový, přepne zisk operačního zesilovače na větší hodnotu.

6.12 Parametry analogově číslicového převodníku

Rychlost vzorkování

Proces vzorkování může být zdrojem značných chyb. Opakovací

kmitočet vzorkování rychlost vzorkování musí být dostatečně vysoký vzhledem k nejvyšší kmitočtové složce vstupního analogového napětí. Na druhé straně vysoký opakovací kmitočet vzorkování způsobuje nadbytečnost informace, což mívá za následek zbytečné náklady na analogově číslicový převodník i zařízení zpracovávající číslicový tvar signálu. Cílem je použít nejnižší opakovací kmitočet vzorkování, který ještě připouští zpracování vstupního analogového napětí bez chyby. Nejnižší přípustný opakovací kmitočet vzorkování je určen Shanonovým - Kotelnikovým vzorkovacím teorémem, který nám říká, že pro nezkreslený přenos je nezbytné přenést více než dva body amplitudy nejvyšší kmitočtové složky analogového signálu. Jinými slovy, nemá-li docházet ke zkreslení analogového signálu, musí být opakovací kmitočet vzorkování f_{opak} vyšší než dvojnásobek nejvyšší kmitočtové složky spektra analogového signálu.

Pokud tento předpoklad není splněn, není zkreslen analogový signál jen potlačením kmitočtových složek vyšších než $f_{opak}/2$, ale tyto vyšší kmitočtové složky analogového napětí ovlivní i kmitočtové složky analogového signálu nižší než $f_{opak}/2$, což plyne ze spektra amplitudově modulovaných impulsů.

Pro další číslicové zpracování nás někdy zajímají jen nízkofrekvenční složky analogového signálu. I v tomto případě musíme volit opakovací kmitočet s ohledem na nejvyšší kmitočtovou složku analogového napětí. Zde však můžeme použít i jiného řešení: kmitočtové spektrum analogového signálu před vzorkováním omezíme vhodnou dolní propustí.

Rychlost převodu

U analogově číslicového převodníku rychlost vzorkování je zpravidla totožná s rychlostí převodu; vyjadřujeme ji počtem úplných převodů za vteřinu. Někdy však je specifikována počtem bitů/s. Často uváděná doba převodu je reciproká hodnota rychlosti převodu.

Rozlišovací schopnost, kvantizační chyba, nelinearita

Rozlišovací schopnost analogově číslicového převodníku je určena počtem úrovní, na něž jsme rozdělili rozsah vstupního

analogového napětí. Jelikož je výstupní slovo převodníku obvykle vyjádřeno číslem v přirozeném dvojkovém kódu, vyjadřuje se často rozlišovací schopnost počtem bitů ve výstupním slovu. Sestává-li např. výstupní slovo z 10 bitů, pak je vstupní rozsah rozdělen na 1023 = 1024 - 1 = 2^{10} -1 diskrétních úrovní. Rozlišovací schopnost je $1/(2^{10}-1)$, kde n je počet bitů ve výstupním slovu. Jelikož obvykle $2^n \gg 1$, je rozlišovací schopnost převodníku $1/2^n$.

Vstupní analogové napětí, které může nabývat libovolné úrovně v mezích vstupního rozsahu, je kvantováno do určitého počtu kvantizačních úrovní. Chybu vzniklou tímto procesem nazýváme kvantizační chyba. Tato chyba může dosahovat maximální hodnoty, která odpovídá $\pm$ 1/2 nejnižšího bitu výstupního slova převodníku. Lze ji zmenšit použitím více bitů ve výstupním slovu převodníku.

Jelikož se snadněji definují polohy přechodů z jedné výstupní úrovně do druhé definujeme a měříme obvykle chyby u analogově číslicových převodníků v jednotkách vstupních analogových hodnot v místech přechodů z jedné úrovně do druhé. První přechod z jedné úrovně na druhou se nemusí vyskytovat přesně na úrovni 1/2 hodnoty odpovídající nejnižšímu bitu ve výstupním slovu; vzniká chyba způsobená napěťovým posunem (obr. 6.31). Dalšími možnými chybami je změna měřítka (chyba zisku, obr. 6.32) a nelinearita analogově číslicových převodníků (obr. 6.33).

Integrální a diferenciální nelinearita analogově číslicového převodníku

Kvantizačních úrovní analogově číslicových převodníků bývá obvykle značný počet. V tomto případě je vhodné rozlišit integrální a diferenciální nelinearitu. Na obr. 6.1 jsme spojili středy kvantizačních úrovní spojnicí a. Je-li tato spojnice přímkou, je přívod lineární.

Integrální nelinearitu N_{int} definujeme normalizovaným rozdílem mezi maximální a minimální strmostí spojnice středů kvantizačních úrovní.

$$N_{int} = \frac{k_{max} - k_{min}}{k_{stř}}$$

kde k_{max}, maximální, k_{min} minimální, $k_{stř}$ střední strmost spojnice, přičemž strmost k je definována vztahem

$$k = \frac{dU_{výst}}{dU_{vstup}}$$

Typická integrální nelinearita desetibitového analogově číslicového převodníku s postupnou aproximací je větší než $\pm 10^{-3}$.

U ideální charakteristiky analogově číslicového převodníku jsou rozdíly mezi jednotlivými dílčími úrovněmi stejně veliké. Velikost těchto "schodů" jsme si označili na obr. 6.1 symbolem s. Jsou-li tyto schody různě veliké, vyskytuje se u příslušného analogově číslicového převodníku diferenciální nelinearita. Diferenciální nelinearíta N_{dif} je definována vztahem

$$N_{dif} = \frac{s_{max} - s_{min}}{s_{stř}}$$

kde s_{max}, s_{min}, $s_{stř}$ je maximální, minimální, střední hodnota schodů.

Diferenciální nelinearita je např. důležitým parametrem analogově číslicového převodníku při statickém vyhodnocování analogového napětí.

Chyba způsobená dobou vzorkování

Doba vzorkování, která je u mnohých typů analogově číslicových převodníků totožná s dobou převodu, může způsobit chybu v převodu. Tato chyba je způsobená změnou vstupního analogového signálu během doby vzorkování. Pro jednoduchost uvažujme převod sinusového napětí

$$u = U_M \sin \omega t$$

a zkoumejme, jaká chyba může nastat vlivem doby vzorkování. Chtějme převést toto harmonické napětí v čase t = 0, tj. při průchodu sinusového napětí nulou obr. 6.34 . V tomto bodě má sinusový průběh maximální strmost, která je určena vztahem

$$\left[\frac{d\,n}{d\,t}\right]_{t=0} = \left[U_M\,\omega\,\cos\,\omega\,t\right]_{t=0} = U_M\,\omega = U_M\,2\pi\,f$$

Aproximujme sinusové napětí kolem bodu t = 0 přímkou

$$u \doteq t\;U_M\,2\pi f$$

Je-li doba vzorkování totožná s dobou převodu $t_{přev}$, pak během této doby nastane změna aproximovaného sinusového napětí

$$\Delta u = U_M\;2\pi\;f\;\;t_{přev}$$

Tento vztah vyjadřuje maximální možnou chybu v převodu sinusového napětí způsobenou dobou převodu. Graficky je uveden na obr. 6.35. Skutečná chyba závisí na typu převodníku (např. zda jednotlivé bity výstupního slova se tvoří současně či od nejvyššího řádu k nejnižšímu řádu).

Z předcházejícího vztahu můžeme určit maximální kmitočet sinusového napětí pro zvolenou přípustnou chybu způsobenou dobou převodu

$$f = \frac{\Delta u}{2\;U_M\;t_{přev}}$$

Chybu analogově číslicového převodníku způsobenou dobou převodu můžeme podstatně zmenšit, zařadíme-li před převodník vzorkovací obvod s analogovou pamětí. Tento obvod odebere vzorek analogového napětí během doby t_{vzor} a hodnotu tohoto vzorku si zapamatuje alespoň po dobu $t_{přev}$. Maximální možná chyba Δu analogově číslicového převodníku při převodu sinusového napětí způsobená dobou převodu se v tomto případě zmenší s faktorem $t_{vzor}/t_{přev}$.

6.13 Kódy používané u číslicově analogových a analogově číslicových převodníků

Přirozený dvojkový kód

U číslicově analogových a analogově číslicových převodníků se nejčastěji používá přirozený dvojkový kód, obvykle se používá vyjádření ve zlomcích. Sestává-li se slovo z n bitů, má nejvyšší bit (nejvyšší dvojkový řád) váhu 2^{-1}, další bit 2^{-2} a nejnižší bit

2^{-n}. Jako příklad jsou uvedeny v tab. 6.3 hodnoty čtyřbitového slova.

Jsou-li všechny bity ve slovu jedničkové, odpovídající číslo je $1 - 2^{-n}$, tj. normalizovaný plný rozsah zmenšený o nejnižší bit. Pro čtyřbitové slovo sestávající ze samých jedniček obdržíme $1 - 2^{-n} = 1 - 2^{-4} = 15/16$.

Pokud bychom chtěli být přesní, měli bychom použít řádovou čárku, tj. psát

$$[0,1111]_2 = [1 - 0,0001]_2$$

V praxi však používáme zápis 1111, což odpovídá v desítkové soustavě 15. Jinými slovy zápis "1111" odpovídá 1111/(1111 + 1) = = 15/16.

V tab. 6.4 jsou uvedeny váhy přirozeného dvojkového kódu. Poznamenejme, že váha nejnižšího dvojkového řádu odpovídá rozlišovací schopnosti; má-li slovo n bitů, vyjadřuje tato tabulka též rozlišovací schopnost.

Kódování napětí obou polarit

Dosud jsme uvažovali analogové napětí jedné polarity, použitý kód vyjadřoval velikost bez ohledu na polaritu. Mnohé číslicově analogové a analogově číslicové převodníky jsou řešeny jen pro jednu polaritu napětí, což v řadě aplikací plně vyhovuje.

Pro vstupní napětí obou polarit u analogově číslicového převodníku je třeba ve výstupním slovu použít kromě hodnotových bitů ještě "znaménkový bit". Podobně je třeba použít znaménkový bit ve vstupním slovu číslicově analogového převodníku, pokud výstupní analogové napětí má měnit polaritu.

Pro vyjádření analogového napětí obou polarit nejčastěji používáme: a) přirozeného dvojkového kódu se znaménkem
b) prvního doplňku
c) druhého doplňku
d) přirozeného dvojkového kódu s napěťovým posunem

Jednotlivé kódy pro slovo sestávající ze 4 bitů s jedním vyhraženým pro znaménka jsou uvedeny v tab. 6.5.

Přirozený dvojkový kód se znaménkem

Amplituda je vyjádřena určitým počtem bitů (v našem příkladu třemi bity), k nimž přidáváme znaménkový bit. Obvykle značí 0, znaménko +, 1 znaménko -. Tento kód je často užíván u číslicově analogových převodníků pracujících v okolí nulového napětí, neboť v tomto kódu se snadno uskuteční přechod z malého kladného na malé záporné napětí a opačně. U tohoto kódu se nemění větší počet hodnotových bitů při přechodu nulovou úrovní.

Přirozený dvojkový kód s napěťovým posunem

Tento kód je nejjednodušší pro převodníky. Všechny bity ve slovu vyjadřují amplitudu: nule v přirozeném dvojkovém kódu je přiřazeno maximální záporné napětí a největšímu číslu maximální kladné napětí. Jinými slovy, chceme-li např. vytvořit čtyřbitový číslicově analogový převodník s rozsahem výstupního napětí -10 V až + 10 V ze čtyřbitového převodníku s rozsahem výstupního napětí převodníku 0 - 20 V, posuneme vstupní napětí převodníku o -10 V. Podobně posouváme vstupní napětí u analogově číslicového převodníku. Jelikož při přechodu nulovým napětím se mění hodnota nejvyššího bitu slova, indikuje nám tento bit též záporné či kladné analogové napětí.

Výhodou přirozeného dvojkového kódu s napěťovým posunem je jednoznačné vyjádření nulového analogového napětí. Převod kódu na jiný je rovněž snadný. Obzvláště jednoduchý je převod na druhý doplněk, který se obvykle používá u mikropočítačů. Provádí se doplňkem nejvyššího bitu slova. Jelikož se v tomto kódu při číslicovém zpracování informací slovo sestávající ze samých nul nepoužívá, můžeme toto slovo použít pro kontrolu nebo nastavení.

Nevýhodou tohoto kódu je změna ve všech dvojkových řádech při přechodu z nulového napětí na nejmenší záporné napětí. Tato změna zpravidla vyvolává nepřesnosti převodníku. Změna analogového napětí v okolí nuly je velmi častá.

Druhý doplněk

Při tomto vyjádření kladná čísla označujeme stejně jako v přirozeném dvojkovém kódu; v nejvyšším bitu je nula.

Vyjádření záporného čísla v druhém doplňku získáme z vyjá-

dření příslušného kladného čísla v přirozeném dvojkovém kódu záměnou nul a jedniček v jednotlivých dvojkových řádech s aritmetickým přičtením jedničky k nejnižšímu dvojkovému řádu. Např. druhý doplněk -3/8 (0011) je 1100 + 0001 = 1101.

V číslicových zařízeních provádějících aritmetické operace, se druhý doplněk velmi často používá, neboť operaci odčítání lze nahradit operací sčítání. Např. 4/8 - 3/8 = 1/8 v druhém doplňku 0100 + 1101 = 0001 (přenos z nejvyššího dvojkového řádu se neuvažuje).

Porovnáním druhého doplňku s přirozeným dvojkovým kódem s napěťovým posunem zjistíme, že obě vyjádření se liší jen v nejvyšším dvojkovém řádu; převod získáme velmi snadno záměnou nul a jedniček v tomto řádu. Jelikož převodníky mají často na vstupu (výstupu) klopné obvody s výstupy Q i $\bar{Q}$ lze obvykle snadno přizpůsobit převodníky pracující v přirozeném dvojkovém kódu s napěťovým posunem k číslicovým zařízením, která používají druhý doplněk.

První doplněk

Používá se (avšak méně než druhý doplněk) v číslicových zařízeních provádějících aritmetické operace. První doplněk záporného čísla lze získat z přirozeného dvojkového kódu téhož, avšak kladného čísla pouhou záměnou nul a jedniček ve všech "neznaménkových" dvojkových řádech. Jinými slovy, první doplněk vytvoříme záměnou nul a jedniček v každém dvojkovém řádu kladné hodnoty pro získání odpovídající záporné hodnoty. To platí též pro nulu, tj. nula je vyjádřena 0000 nebo 1111. Např. první doplněk 0011 je 1100. Pomocí prvního doplňku převádíme odčítání na sčítání tím, že vytvoříme první doplněk menšitele, který přičteme k menšenci a k výsledku přičteme 1 v nejnižším dvojkovém řádu, což nazýváme kruhový přenos, neboť přičtení této jedničky je odvozeno z přenosu v nejvyšším řádu. Přenos z nejvyššího řádu dále neuvažujeme. Např. pro rozdíl 4/8 - 3/8 můžeme psát 0100 + 1100 = 0000 plus 0001 (kruhový přenos) = 0001, což odpovídá 1/8.

Nevýhodou prvního doplňku je dvojí vyjádření nuly a tím obtížnější převod na jiný kód ve srovnání s druhým doplňkem. Vyjadřujeme-li záporná čísla pomocí prvního doplňku, lze užít čísli-

cově analogový převodník pracující v druhém doplňku tak, že před převodník zařadíme sčítací obvod (sčítačku). Tento sčítací obvod přičte jedničku k nejnižšímu dvojkovému řádu v případě, že v nejvyšším dvojkovém řádu je jednička, která indikuje, že se jedná o záporné číslo. Můžeme však použít i jiný způsob; k výstupnímu analogovému napětí přičteme analogové napětí odpovídající nejnižšímu dvojkovému řádu vstupního slova za předpokladu, že v nejvyšším dvojkovém řádu (bitu) vstupního slova je jednička. To můžeme uskutečnit odporovým vydělením napětí, které odpovídá nejvyššímu dvojkovému řádu (bitu) a přičtením vyděleného napětí k výstupnímu analogovému napětí.

7. TECHNICKÉ PROSTŘEDKY PRO NÁVRH A OŽIVOVÁNÍ MIKROPOČÍTAČOVÝCH SYSTÉMŮ

Vývoj systému s mikroprocesorem bez prostředků, které tento vývoj usnadňují, je obtížně schůdná cesta.

Konstruktér systému musí obvykle řešit v souvislostí s vývojem uživatelského programu následující dílčí úlohy:

- sestavení programu
- zkoušení programu bez ohledu na okolí
- zkoušení programu se simulací okolí
- zavedení programu do mikropočítačového systému
- zkoušení programu v mikropočítačovém systému
- zkoušení zapojení s integrovanými obvody vysoké integrace

Pro jednotlivé činnosti se používají různé vývojové prostředky, např. mikropočítačové vývojové systémy (které jsou určeny pro vývoj uživatelských programů), nepřímé programy (které umožňují vývoj a zkoušení uživatelských programů ve větších hostitelských počítačích), logické analyzátory (které slouží především pro zkoušení technického vybavení).

4.1 Mikropočítačový vývojový systém

Mikropočítačový vývojový systém je zařízení obsahující všechno nezbytné technické i programové vybavení pro vývoj prototypu mikropočítačového systému. V současné době je prakticky nezbytným zařízením při profesionálním vývoji systémů s mikroprocesory.

Tento systém podstatně usnadňuje vývoj mikropočítačového systému. Obsahuje všechno technické vybavení (hardware), které se bude vyskytovat ve vyvíjeném mikropočítačovém systému a dále programovací vývojové prostředky (někdy nazývané pomocné programy nebo

vývojové programy), které usnadňují vývoj nového mikropočítačového systému. Typický mikropočítačový vývojový systém obsahuje:

- mikroprocesor
- paměť s libovolným výběrem
- zdroj hodinových impulsů
- vstupy a výstupy včetně vazebních obvodů (např. dálnopis, zobrazovací jednotku s katodovou obrazovkou a klávesnicí)
- řídicí panel (obsahuje přepínače pro nulování, zastavení, krokování, naplnění registrů atd., dále zobrazovací prvky - elektroluminiscenční diody - pro zobrazení obsahů zvolených registrů)
- vnější paměť (paměť s magnetickým pružným diskem)
- programátor (programovací zařízení) programovatelné permanentní paměti
- programovací vývojové prostředky (pomocné programy) buď na děrné pásce nebo v permanentní paměti.

Mikropočítačový vývojový systém je určen pro zjednodušení nejčastěji se vyskytujících problémů při vývoji nového mikropočítačového systému, jak v oblasti technického, tak v oblasti programového vybavení, pomocí standardní konfigurace, která může být přizpůsobena určité aplikaci jak v oblasti technického, tak v oblasti programového vybavení. Je obvykle řešen modulárně v jedné skříňce s možností rozšíření vložením dalších zapojených desek plošných spojů. Nejčastěji se tato doplnění nebo rozšíření vztahují na možnost zvětšení počtu vstupu i výstupů, rozšíření systému přerušení a rozšíření kapacity operační paměti.

Programátor programovatelné permanentní paměti je prakticky autonomní technické zařízení, které umožňuje programování programovatelné permanentní paměti.

Obvykle umožňuje výpis programu (program listing), zápis programu pomocí klávesnice a kopírování z jedné programovatelné permanentní paměti do druhé.

Skupinové schéma typického mikropočítačového vývojového systému s jednotkou pro propojení s vyvíjeným systémem je na obr. 7.1. Paměť s libovolným výběrem je řešena často po modulech 16 Kbytů s maximální kapacitou 64 Kbytů.Část této paměti je někdy nahrazena reprogramovatelnou permanentní pamětí. Z operační paměti je vyčleněna kapacita cca 2 Kbytů pro monitor; je uložen

v permanentní paměti. Mikropočítačový vývojový systém je vybaven víceúrovňovým systémem přerušení, umožňuje připojení podsystému vstup - výstup a podsystému pro blokový přenos. Mnohdy umožňuje zápis do programovatelné permanentní paměti.

Mikropočítačový vývojový systém mívá ve výbavě následující pomocné programy:

a) Editor, jehož pomocí může uživatel v určitém kódu (obvykle ACII) zapsat zdrojový program do paměti mikropočítače a případně ho opravit.

b) Makroasembler, který kromě překladu provádí výpis použitých symbolů a výpis syntaktických chyb. Tento makroasembler je zpravidla absolutní, to znamená, že zavádí cílový program do operační paměti od zadané adresy. U některých novějších mikropočítačových vývojových systémů se používá přemísťující makroasembler.

c) Kompilátor, zpravidla pro jazyk PL/M, výjimečně pro FORTRAN, COBOL nebo BASIC.

d) Ladicí program (debuger), který rozšiřuje možnosti monitoru při ladění cílového programu. Pomocí emulačního a zkušebního adaptoru je funkce ladicího programu podstatně rozšířena.

e) Vazební program (linker), který vytvoří úplný program z jednotlivých podprogramů ve strojovém kódu. Tyto podprogramy mohou být buď z knihovny podprogramů nebo produktem překladu. Předpokladem je, že podprogramy jsou adresovány relativně, tj. jsou získány pomocí přemísťujícího asembleru nebo kompilátoru. Použití vazebního programu předpokládá zpravidla vybavení mikropočítačového vývojového systému pamětí s magnetickým pružným diskem.

f) Zaváděcí program (loader), který zavádí cílový program do operační paměti od zadané počáteční adresy.

Některé uvedené pomocné programy bývají sjednocovány monitorem. Monitor zpravidla také obsahuje podprogramy umožňující uživateli komunikaci s mikropočítačem pomocí dálnopisu nebo zobrazovací jednotky a též obsahuje podprogramy pro využívání přídavných pamětí.

7.1.1 Emulační a zkušební adaptor

Emulační a zkušební adaptor (in-circuit-Emulator) umožňuje účinně zkoušet uživatelský program a uživatelem vyvíjený systém s mikroprocesorem. Prvně byl použit u mikropočítačového vývojového systému MDS 800 (Intel), nyní se používá i u jiných typů mikropočítačových vývojových systémů.

Při použití emulačního a zkušebního adaptoru se nahradí mikroprocesor ve vyvíjeném systému mikroprocesorem, jenž je součástí mikropočítačového vývojového systému. Tím mohou být kontrolovány procesy probíhající ve vyvíjeném systému. Příklad skupinového uspořádání emulačního a zkušebního adaptoru (MDS 800) je obr. 7.2. Emulační a zkušební adaptor obsahuje vlastní mikroprocesor (emulační mikroprocesor), který je napojen na systémovou sběrnici MDS 800. Přímé řízení vychází přes řízení emulace a emulační program. Mikroprocesor ve vyvíjeném systému je nahrazen svorkou, která je propojovacím kabelem spojena s emulačním a zkušebním adaptorem.

Uživatelský program může být proveden pomocí emulačního a zkušebního adaptoru. Uživatel má možnost provést program, který je uložen buď v paměti mikropočítačového vývojového systému MDS nebo v paměti uživatelem vyvíjeného systému. Uživatel má též možnost pomocí programu volit buď vstupní-výstupní podsystém MDS 800 nebo vstupní-výstupní podsystém obsažený ve vyvíjeném systému. To umožňuje téměř úplně vyzkoušet uživatelský program, i když vyvíjený systém není obvodově úplně sestavený. Tato možnost se často využívá zvláště v případech, kdy se ve vyvíjeném systému použije permanentní paměť. V tomto případě není třeba použít pro zkoušení programovatelnou nebo reprogramovatelnou paměť, ale vyvíjený systém lze vyzkoušet pomocí paměti s libovolným výběrem a po vyzkoušení její obsah zapsat do permanentní paměti. Je třeba si však uvědomit, že provedení programu pomocí mikropočítačového vývojového systému MDS 800 trvá asi o 30 % déle, tj. v tomto případě činnost se neuskutečňuje v reálném čase. To může vést v některých případech, zvláště u časově kritických programů, k chybám. V těchto případech je nezbytné zkoušet navrhovaný systém s vlastní pamětí.

Sledovací obvod (tracer) emulačního a zkušebního adaptoru

umožňuje výpis obsahu pamětí zvláště v bodech větvení programu, čímž se usnadňuje sledování průběhu programu.

Některé mikropočítačové vývojové systémy (např. MDS) umožňují zkoušení vyvíjených systémů s jiným typem mikroprocesoru než obsahuje samotný mikropočítačový vývojový systém. To však umožňují jen ty systémy, které obsahují dva mikroprocesory.

7.2 Postup při vývoji programu pomocí vývojového mikropočítačového systému

Typický postup při vývoji programu pomocí vývojového mikropočítačového systému je na obr. 7.3. Ručně napsaný zdrojový program v jazyku symbolických adres se zapisuje do mikropočítačového vývojového systému klávesnicí dálnopisu nebo zobrazovací jednotky pomocí programu editor, který řídí vstup, umožňuje opravu a ukládá zdrojový program na vhodné medium, např. děrnou pásku. Tento zdrojový program v jazyku symbolických adres se přeloží do cílového programu (strojový jazyk) a zpravidla napíše na děrnou pásku. V této fázi může být též pro kontrolu prováděn výpis programu obsahující zdrojový i cílový program.

Má-li být tento program uložen do paměti s libovolným výběrem mikropočítače, je třeba tento krok provést pomocí zaváděcího programu (loader). Tento program čte adresy paměti a instrukce cílového programu psaného na děrné pásce. Po zavedení cílového programu do operační paměti se tento program provádí za částečné nebo úplné kontroly ladicího programu (debugger program) s cílem odhalení, případně provedení nezbytných oprav. Jsou-li chyby větší, je třeba se vrátit do bodu 1 na obr. 7.3.

Na konci bodu 2 je třeba se rozhodnout, která část programu bude zapsána do permanentní paměti programované maskou nebo do programovatelné či reprogramovatelné permanentní paměti. Pro nastavení adres jednotlivých instrukcí programu v permanentní paměti je možné opakovat postup od bodu 1.

Postup v bodě 4 končí zapsáním cílového programu na děrnou pásku, která slouží pro programovací zařízení programovatelné nebo reprogramovatelné permanentní paměti.

7.3 Ladění programu pomocí mikropočítačového vývojového systému

I když je program přeložen asemblerem bez indikace syntaktických chyb ve výpisu, může program ještě obsahovat chyby, které způsobí chybné řešení úlohy. Proces, kterým se tyto chyby odstraňují, nazýváme laděním a usnadňuje jej ladicí program.

Ladicí program umožňuje zkontrolovat a změnit obsah každé paměťové buňky operační paměti nebo registru pomocí dálnopisu. Umožňuje též provést výpis kterékoliv části programu. Zpravidla umožňuje vyhledat určité slovo v paměti a zrušit nebo vložit určitou instrukci.

Obvykle je ladicí program umístěn na nejvyšších adresách operační paměti a laděný program je umístěn na předem určených adresách. Operátor pouze zapisuje na dálnopisu řídicí příkazy (řídicí sekvence) a program "odpovídá". Např. zapsáním D 75 dáváme příkaz k výpisu (zobrazení) obsahu paměťové buňky mající adresu 75. Je-li obsah této buňky 15_{16}, dálnopis zapíše D 75 15.

Některé mikropočítačové vývojové systémy používají pro ladění ovládací panel mikropočítače místo dálnopisu nebo zobrazovací jednotky. V tomto případě bývá obsah zvoleného registru nebo paměťové buňky zobrazen pomocí elektroluminiscenčních polovodičových diod. Přepínači na ovládacím panelu můžeme spustit nebo zastavit program v libovolném předem zvoleném místě nebo jej můžeme krokovat.

Ladění pomocí dálnopisu je obvykle výhodnější, neboť z výpisu můžeme snadněji sledovat více kroků.

7.4 Simulátor permanentní paměti

U malých mikropočítačů se pro záznam programu používá permanentní paměť. Důvodem je především zachování informace při ztrátě napájecího napětí. Důvodem vývoje však není výhodné použít permanentní paměť programovanou maskou (ROM), programovatelnou permanentní paměť (PROM) nebo reprogramovatelnou permanentní paměť (EPROM), protože jejich obsahy nelze změnit, nebo u posledního typu jen obtížně.

Nemá-li uživatel k dispozici emulační a zkušební adaptér nebo nevyhovuje-li jeho použití z hlediska činnosti v reálném čase, je

výhodné použít simulátor permanentní paměti. Při jeho použití je nahrazena permanentní paměť v uživatelem vyvíjeném systému svorkou připojenou kabelem k paměti s libovolným výběrem, která je součástí simulátoru permanentní paměti. Informace v paměti s libovolným výběrem, odpovídající informacím v permanentní paměti a mohou být podle potřeby uživatelem měněny. Kapacita paměti bývá 1 až 2 Kbytů. Delší programy musí být zkoušeny po částech.

Simulátor permanentní paměti může být součástí mikropočítačového vývojového systému; v tomto případě řízení a vkládání informací do paměti se provádí pomocí tohoto vývojového systému. Pokud simulátor tvoří samostatný přístroj, obsahuje panel, pomocí něhož lze vkládat instrukce do paměti a libovolným výběrem, nebo je mazat, případně umožňuje "zastavení" na zvolené adrese. Tím je možné ladit jednoduché systémy s mikroprocesorem, jejichž délka programu nepřesahuje 200 bytů.

Pomocí některých typů simulátorů permanentní paměti lze též provádět zápis do programovatelné paměti.

7.5 Nepřímé programy

Většina výrobců mikroprocesorů poskytuje nepřímé programy (cross-software). Jednotlivé programy patřící do této kategorie jsou často psány v jazyku FORTRAN a mohou být provozovány na větších počítačích, včetně minipočítačů.

Nepřímé programy obsahují v každém případě nepřímý asembler (cross-assembler), který přeloží zdrojový program napsaný v jazyku symbolických adres do strojového kódu. Mnohdy obsahují nepřímé programy nepřímý kompilátor (cross-compilator) a nepřímý simulátor, (cross-simulator). Nepřímý kompilátor přeloží program vyššího programovacího jazyka (např. PL/M) do strojového kódu. Některé nepřímé kompilátory generují program v jazyku symbolických adres. V tomto případě může programátor v případě potřeby upravit (optimalizovat) program v jazyku symbolických adres z hlediska požadavku na kapacitu paměti, případně z hlediska rychlosti. Upravený program pak asembler přeloží do strojového kódu.

Pomocí nepřímého simulátoru může uživatel, podobně jako pomocí mikropočítačového vývojového systému, vyzkoušet uživatelský

program. Zkoušení však se neuskutečňuje v reálném čase.

Nepřímé programy poskytují následující výhody:

- použití rozmanitých přídavných zařízení hostitelského počítače (např. tiskárny, zobrazovací jednotky), které obvykle mikropočítačový vývojový systém neobsahuje.
- použití operačního systému hostitelského počítače a tím využití jeho vnější velkokapacitní paměti.
- použití nepřímých programů pro různé mikropočítače na jednom typu hostitelského počítače.
- po získání nepřímých programů jsou náklady na zkoušení uživatelských programů zpravidla nižší ve srovnání s použitím mikropočítačového vývojového systému.

Někdy vznikají problémy při přenosu uživatelských programů získaných na hostitelském počítači do uživatelského mikropočítačového systému. Větší počítače často používají jako nosičů informací magnetické pásky nebo magnetické disky; mikropočítače používají děrnou pásku, magnetickou kazetu, magnetický pružný disk nebo jen permanentní paměť. Přímý přenos nosičů informací mezi nimi není obvykle možný. Pro usnadnění přenosu uživatelských programů, resp. pro zkoušení programů v uživatelském systému se často jako mezičlánek ještě používá mikropočítačový vývojový systém s emulačním a zkušebním adaptorem, případně simulátor permanentní paměti.

7.6 Školní mikropočítače

Školní mikropočítač, který též nazýváme elementární mikropočítačový vývojový systém, nebo mikropočítačový kit, je úplný a funkčně schopný mikropočítač. Obvykle obsahuje následující díly:

- mikroprocesor
- paměť s libovolným výběrem RAM s kapacitou 0,5 až 16 KB pro zápis uživatelského programu
- místo pro programovatelnou nebo reprogramovatelnou permanentní paměť, které využije uživatel
- jednotku vstup-výstup
- svorkovnici pro připojení dálnopisu nebo zobrazovací jednotky s klávesnicí

- svorkovnici pro připojení magnetické kazetové paměti.

Programové vybavení tohoto mikropočítačového vývojového systému je poměrně jednoduché; obsahuje obvykle jen monitor, který zaujímá kapacitu 1 až 2 KB. Pomocné programy, např. editor a asembler zpravidla neobsahuje.

Monitor umožňuje uživateli komunikací s počítačem obvykle pomocí dálnopisu. Dále obvykle umožňuje:

- čtení a děrování děrné pásky
- přenos obsahu určité části operační paměti na jiné místo paměti, přičemž záznam v původní části paměti může být zachován
- zobrazení nebo vytištění určité části operační paměti
- změnu obsahu operační paměti
- zobrazení a změnu obsahu vnitřních obecných pracovních registrů mikroprocesoru
- přerušení programů na určité adrese
- spuštění programu na zvolené adrese.

7.6.1 Oblasti využívání

Použití školních mikropočítačů pro profesionální vývoj mikropočítačů systémů s mikroprocesory je sporné. Je třeba si uvědomit, že při sestavě programu se značná doba stráví zkoušením a laděním programu. Právě pro tyto činnosti nejsou zpravidla školní mikropočítače dostatečně vybaveny.

Principiálně je vývoj programu možný na každém školním mikropočítači. Není však rozumné řešit vývoj rozsáhlejších systémů jeho prostřednictvím. Je však účelné mít školní mikropočítač pro zkoušení kratších programů.

Těžiště použití školních počítačů leží v jejich využití pro výuku jak individuální, tak kolektivní v kursech. V těchto případech bývá často relativně drahý dálnopis určený pro komunikaci s uživatelem nahrazen jednoduchou klávesnicí s jednoduchou zobrazovací jednotkou tvořenou polovodičovými elektroluminiscenčními diodami.

Při vývoji mikropočítače pro určité použití, zvláště pro větší výrobní série, je někdy účelné porovnat vhodnost různých typů

mikropočítačů, zvláště z hlediska potřebné kapacity paměti - pro tento účel lze využít příslušný typ školních mikropočítačů. Není-li předem známo, zda výpočetní mohutnost určitého typu mikropočítače bude dostačující pro uvažovaný účel, je též mnohdy výhodné použít školní počítač. Tímto typem mikropočítače může být ověřena struktura vstupů-výstupů a struktura přerušení. Při malosériové výrobě zařízení, jehož součástí má být mikropočítač, je někdy ekonomicky výhodné v zařízení přímo použít školní mikropočítač.

7.6.2 Vlastnosti školních mikropočítačů

Obsluha školních mikropočítačů není samozřejmě tak snadná, jako několikanásobně dražších mikropočítačových vývojových systémů. Instrukce a data se obvykle zadávají v šestnáctkové soustavě. To samozřejmě předpokládá, že uživatel instrukce v této soustavě zná, resp. program napsaný v jazyku symbolických adres nejprve ručně přeloží do šestnáctkové soustavy pomocí seznamu instrukcí. Jen výjimečně lze u některých školních mikropočítačů vkládat program pomocí mnemotechnických zkratek.

Školní mikropočítače jsou obvykle vybaveny:

- jednoduchou klávesnicí, která slouží jako vstup
- číslicovým ukazatelem, který slouží jako výstup pro komunikaci s uživatelem
- programem v permanentní paměti, který umožňuje se zařízením pracovat.

Počet tlačítek bývá přibližně 25 (16 tlačítek pro šestnáctkové číslice, ostatní jsou funkční tlačítka). Číslicový ukazatel je často tvořen sedmisegmentovými ukazateli. V permanentní paměti jsou uloženy programy provádějící obslužné funkce.

Pomocné programy pro zadávání jednotlivých informací i čtení jednotlivých paměťových buněk obsahují všechny školní mikropočítače. Také obsahují tlačítko "na základní adresu", které nastaví mikropočítač do definovaného základního stavu. Dále obsahují tlačítko "krokování", které umožňuje řešení programu po jednotlivých instrukcích. Některé tyto mikropočítače jsou vybaveny programem a svorkovnicí pro připojení vnější, např. kazetové paměti.

7.6.3 Vývoj programu pomocí elementárního mikropočítačového vývojového systému

Uživatel napíše program zpravidla v jazyku symbolických adres pro zvolený mikroprocesor. Jelikož elementární mikropočítačový vývojový systém není obvykle vybaven asemblerem, je třeba zdrojový program v jazyku symbolických adres ručně přeložit do strojového jazyka. Program se ve strojovém jazyku v tomto případě zapisuje často v šestnáctkové soustavě. Tento program se zavádí do operační paměti od určené počáteční adresy pomocí monitoru a tlačítka "vstup".

Program ve strojovém kódu zavedený do systému může být laděn pomocí monitoru a případné chyby mohou být opravovány přímo ve strojovém kódu. Jelikož však se překlad provádí ručně a ladicí možnosti monitoru jsou omezené, je ladění uživatelského programu tímto postupem značně náročné, zvláště v případech, je-li program delší než 1 Kbyte. V případě, že školní mikropočítač obsahuje ladicí program a asembler (což bývá zřídka), je vývoj programu značně usnadněn.

7.7 Logické analyzátory

Nárůst využívání a složitosti číslicových systémů si vynutil konstrukci nového přístroje, "logického analyzátoru", pomocí kterého je konstruktér schopen detekovat chybu a určit její místo. Logický analyzátor se stává nezbytným přístrojem při konstrukci a opravě složitějších číslicových systémů, podobně jako běžný osciloskop při konstrukci a oživování elektronických systémů, u kterých je třeba sledovat průběh napětí v reálném čase. Někteří odborníci předpovídají, že během několika málo let bude se v laboratoři častěji vyskytovat logický analyzátor než osciloskop.

Logický analyzátor je přístroj pro sběr a zobrazení informací v číslicové doméně. (Osciloskop zobrazuje v časové doméně, spektrální analyzátor ve frekvenční doméně). Informace v číslicové doméně sestávají z dat, která se vyskytují jako sled logických hodnot 0,1 zpravidla současně na několika vodičích.

Logický analyzátor neanalyzuje data, jak by mohl název naznačovat, ale jen je zobrazuje ve tvaru, umožňujícím uživateli deteko-

vat chyby. Jen některé typy výjimečně mají vstup pro přenos uložených dat do počítače, který analýzu provádí.

Logické analyzátory umožňují současně sledovat logické hodnoty v několika místech současně; jsou obvykle čtyř až šestnáctikanálové, některé mají až 32 kanálů. V každém kanálu je paměť, která umožňuje uchovat 12 až několik tisíc bitů. Tato paměť, kterou si můžeme představit jako posuvný registr, umožňuje zaznamenat a zobrazit i neopakovatelný (neperiodický) sled dat, nebo zobrazit sled dat, který předchází synchronizaci.

Zápis do posuvných registrů je synchronizován hodinovými impulsy zkoušeného zařízení, nebo vnitřními hodinovými impulsy logického analyzátoru. Data se neustále zapisují do paměti; naplní-li se paměť, přepisují se dříve zapsaná data novými.

7.7.1 Skupinové schéma a funkce logického analyzátoru

Skupinové schéma logického analyzátoru s 8 kanály je na obr. 7.4. Některé logické analyzátory mají na jednotlivých vstupech aktivní elektronické sondy, pomocí nichž se omezí impedanční zatížení měřených bodů vyšetřovaného systému. U některých typů je problém zatížení měřených bodů řešen pomocí "vícenásobné sondy", z které vycházejí již jen krátké spoje k měřeným bodům.

Jednotlivé vstupní analogové signály jsou po zesílení vedeny na napěťové komparátory. Většina logických analyzátorů mění vstupní analogové impulsy na logické hodnoty 0,1 pomocí jednoho prahového napětí. Toto prahové napětí je volitelné uživatelem podle typu vyšetřovaných logických obvodů. Některé logické analyzátory mají dvě prahová napětí, jedno se nastaví na minimální úroveň logické hodnoty 1, druhé na maximální úroveň logické hodnoty 0 (uvažujeme kladnou logiku). Tím lze detekovat chyby či poruchy např. parazitní impulsy, zákmity přechodů impulsů, pomalé náběžné hrany a vzájemné zpoždění impulsů.

Vzorky dat se ukládají do posuvných registrů. V závislosti na kapcitě posuvných registrů, která bývá 12 až 2048 bitů, je stále zaznamenáno 12 až 2048 posledních vzorků dat. Hodinový kmitočet posuvných registrů, tj. rychlost vzorkování vstupních dat, může být odvozen interně v logickém analyzátoru, nebo z vnějšího zdroje

(hodinového kmitočtu zkoumaného uživatelského systému).

Generátor zobrazení mění data uložená v posuvných registrech na signály, které je možné použít pro zobrazení. Zobrazovací jednotka je obvykle tvořena katodovou obrazovkou, která někdy bývá součástí logického analyzátoru (logický analyzátor je samostatný přístroj), někdy se využívá obrazovka v oscilografu (sdružený přístroj oscilograf - logický analyzátor).

Synchronizace (trigger) logického analyzátoru se podstatně liší od synchronizace oscilografu. U logického analyzátoru se vzorky vstupních signálů trvale zapisují do posuvných registrů. Tento proces lze v určitém časovém okamžiku přerušit a tento časový okamžik je určován synchronizačním signálem, od kterého započne stabilní zobrazení dat. Zobrazují se data uložená do posuvných registrů během času, který byl zvolen uživatelem.

Logické obvody synchronizace obvykle umožňují:

a) Přerušení záznamu synchronizačním signálem. V posuvných registrech zůstanou zaznamenána data, která předcházela synchronizačnímu signálu; tato data se též zobrazí.

b) Předvolbu, kolik dat vstupních stavů následujících za synchronizačním signálem má být ještě zaznamenáno, než nastane přerušení záznamu. Tím se umožňuje uživateli pozorovat zvolený počet stavů, který následuje za synchronizačním signálem. Tento počet stavů lze zvyšovat, až všechna data zaznamenávaná do posuvných registrů leží za synchronizačním signálem.

U některých typů logických analyzátorů může uživatel volit zpoždění záznamu za synchronizačním signálem. Např. uživatel má možnost zastavit zápis do posuvných registrů až po 10000 vstupních stavech.

Synchronizační signál u logického analyzátoru může být vnější, nebo vnitřní. Vnitřní synchronizační signál může být odvozen z jednoho vstupního signálu. Často však je odvození vnitřního synchronizačního signálu ze změny logické hodnoty v jednom kanálu nedostačující pro specifikaci požadovaného okamžiku. Proto lze pro odvození vnitřního synchronizačního signálu obvykle použít libovolně zvolenou kombinaci hodnot sledovaných vstupních signálů. Tento způsob synchronizace nazýváme "předvolba slova" (word recognition);

umožňuje ji prakticky každý logický analyzátor. Při tomto způsobu synchronizace je synchronizační impuls odvozen z ekvivalence sledovaného vstupního slova dat a předvoleného specifického slova. Tuto předvolbu provede operátor a může např. vyjadřovat chybovou podmínku nebo určitou instrukci.

U některých logických analyzátorů máme možnost použít tzv. "podmíněnou synchronizaci" (qualifier). Vznik synchronizačního impulsu je podmíněn jednak ekvivalencí sledovaného vstupního slova dat a předvoleného specifického slova a ještě další podmínkou, např. určitou minimální amplitudou hodinových impulsů.

7.7.2 Záznam sledovaných dat do paměti

Sled dat, který vstupuje do logického analyzátoru, se neustále mění. Pro určení logických hodnot, které mají být zapsány jako jednotlivé bity v posuvných registrech, je sled vstupních dat periodicky vzorkován hodinovými impulsy.

Odebírá-li se vzorkovací kmitočet ze zkoušeného systému, mluvíme o synchronním vzorkování. Logický analyzátor ukládá do posuvných registrů změny logických hodnot jen během vnějších hodinových impulsů (resp. během nástupní nebo sestupné hrany hodinových impulsů).

Data (stavy) ukládaná do posuvných registrů (a též zobrazovaná) odpovídají stavové tabulce zkoušeného systému, nemusí však odpovídat skutečnému sledu logických hodnot 0 a 1 na vstupech jednotlivých kanálů.

Pro vyšetřování skutečného sledu logických hodnot (včetně parazitních impulsů) na vstupu jednotlivých kanálů logického analyzátoru se vstupní informace vzorkují asynchronními hodinovými impulsy generovanými uvnitř logického analyzátoru, které nejsou v relaci s hodinovými impulsy zkoumaného systému. Obvykle je opakovací kmitočet asynchronních hodinových impulsů podstatně vyšší než opakovací kmitočet zkoumaného systému. To umožňuje podrobné studium signálů na vstupech jednotlivých kanálů (šíře impulsů, zpoždění mezi jednotlivými kanály, zákmity).

Časová rozlišovací schopnost (nejkratší detekovaný čas mezi přechody logických hodnot) je roven vzorkovacímu intervalu a době

ustálení potřebné pro detekci a uložení nové logické hodnoty. Např. je-li opakovací kmitočet asynchronních hodinových impulsů 100 MHz, tj. vzorkovací interval je 10 ns, a doba ustálení 5 ns, má logický analyzátor časovou rozlišovací schopnost 15 ns.

Na obr. 7.5 je jako příklad uvedeno zobrazení při vzorkování synchronními a rychlými asnychronními hodinovými impulsy. Při synchronním vzorkování se ukládají do paměti a zobrazují jednotlivé stavy vstupních dat. Při rychlém asynchronním vzorkování lze studovat i časové vztahy. Např. je zřejmé, že poslední přechod z logické hodnoty 1 do hodnoty 0 v 1. kanálu nastává později než příslušný obrácený přechod v 2. kanálu. Při tomto způsobu vzorkování se též detekuje parazitní impuls v kanálu 2.

Některé logické analyzátory mohou pracovat v módu s pomocnou pamětí (mode latch), který umožňuje především zjistit výskyt parazitních impulsů i při relativně nízké rychlosti vzorkování a záznamu vstupních signálů. Je třeba si uvědomit, že zápis vstupních dat do posuvných registrů se zpravidla provádí během hrany (nástupní nebo sestupné) hodinových impulsů. Při tomto způsobu zápisu se však ztrácí informace o všech změnách signálu, které se vyskytují mezi následnými hranami hodinových impulsů. Pokud je zkoušený obvod (uživatelský systém) synchronní, nemají tyto změny signálu mezi hranami hodinových impulsů vliv na funkci uživatelského systému. Logické analyzátory, které mají možnost pracovat v módu s "pomocnou pamětí", umožňují pozorovat tyto změny signálu. Při tomto módu činnosti se i malý impuls vyskytující se mezi následnými hranami hodinových impulsů nejprve zaznamenává do pomocné paměti a s následující hranou hodinových impulsů se zaznamená do posuvného registru. Tímto způsobem sice nelze zjistit přesnou dobu výskytu tohoto parazitního impulsu, lze ho však identifikovat, což je důležité u asnychronních logických obvodů, resp. u asynchronních vstupů a výstupů mikropočítačů.

7.7.3 Paměť

Posuvné registry jednotlivých kanálů logického analyzátoru bývají obvykle z ekonomického důvodu realizovány jako paměť s libovolným výběrem. Požadujeme nejen dostatečný počet kanálů, ale též dostatečnou kapacitu paměti příslušející jednomu kanálu.

Kapacita paměti s libovolným výběrem musí být rozdělena na určitý počet kanálů, z nichž každému přísluší určitý počet bitů. Paměť o kapacitě 256 bitů může být rozdělena na 16 kanálů při 16 bitech/kanál, nebo na 8 kanálů při 32 bitech/kanál. Výhodnost jednoho nebo druhého uspořádání závisí na konkrétní aplikaci. Např. pro vyšetřování sériového přenosu dat mezi přístroji je pravděpodobně výhodné použít formát mající 4 kanály při 1024 bitech/kanál; zatímco při paralelním přenosu dat a stejné celkové kapacitě paměti 4096 bitů je lepší použít formát 16 kanálů při 256 bitech/kanál. Některé logické analyzátory umožňují volbu formátu, tj. umožňují celkovou kapacitu paměti rozdělit na zvolený počet kanálů.

7.7.4 Zobrazování

Data uložená v paměti mohou být zobrazována následujícími způsoby obr. 7.6 a obr. 7.7 :

- sledem logických hodnot 0 - 1
- tabulkou logických hodnot
- časovým diagramem
- maticově, mapou

Při zobrazení pomocí časového diagramu jsou zobrazovány na stínítku katodové obrazovky jednotlivé kanály ve tvaru časových průběhů. Je třeba si však uvědomit, že tyto časové průběhy jsou idealizovanou aproximací skutečných vstupních signálů, které jsou na vstupu jednotlivých kanálů. Jelikož nemůžeme sledovat skutečné časové průběhy jednotlivých kanálů logický analyzátor nevykonává funkci vícekanálového oscilografu , mohou některé parazitní impulsy chybět v časovém diagramu na stínítku. Určitou variantou zobrazení pomocí časového diagramu je vyjádření úrovní v jednotlivých kanálech sledem logických hodnot 0 - 1.

Tabulka logických hodnot vyjadřuje logické hodnoty přísluš ející jednotlivým kanálům v dobách zápisu do posuvných registrů. Tabulka logických hodnot bývá vyjádřena v osmičkové nebo šestnáctkové soustavě.

Při zobrazování mapou, které se nazývá též maticové zobrazení, je představováno každé slovo určitým bodem na stínítku.

Zobrazované slovo, jehož délka je n bitů, je rozděleno na dvě poloviny. Vyšší bity určují vertikální polohu bodu, nižší bity slova určují horizontální polohu bodu. Polohy bodů pro určitý sled dat vytvářejí mapu, kterou lze využít pro rychlé celkové vyhodnocení dat bez detailní analýzy. Pro zjištění případného zdroje chyby je třeba provést detailnější analýzu, např. pomocí pohyblivého kurzoru v mapě, nebo jiného způsobu zobrazování.

7.7.5 Použití logických analyzátorů, výrobci, cena

Logické analyzátory se nejčastěji používají při hledání chyb při připojování přídavných zařízení k mikropočítačovému systému, zvláště při asynchronním přenosu dat.

Mapa umožňuje v omezené míře zkoušet program. Připojíme-li např. 16 bitovou adresovací sběrnici mikroprocesoru na logický analyzátor a pozorujeme-li stavy na této sběrnici, získá se obraz o průběhu programu. Zjišťují se též chybné adresy, které leží mimo rozsah použité paměti.

První logické analyzátory byly dány na trh v roce 1973. V současné době je vyrábí řada firem, z nichž nejznámější jsou: Hewlett Packard, Biomation, Tektronix. V Evropě logické analyzátory vyrábí dosud jen jedna firma (Dolch Logic Instruments, NSR).

Některé typy logických analyzátorů jsou uvedeny v tab. 7.1.

Ceny se pohybují obvykle v rozmezí 4000 až 14000 $; cena nejjednodušších typů je cca 600 $.

7.7.6 Mikropočítačové analyzátory

Některé logické analyzátory novějšího provedení určené pro zobrazování stavů mikroprocesoru jsou nazývány mikropočítačovými analyzátory. Zobrazují instrukce, adresy a logické hodnoty na dalších vodičích mikropočítače. Příkladem mikropočítačového analyzátoru je Hewlett-Packard 1611.

Mikropočítačové analyzátory obvykle zobrazují jednotlivé instrukce programu mnemotechnicky, jsou zpravidla orientovány pro systémy s určitými typy mikroprocesorů nejčastěji I 8080, M 6800 .

Mikropočítačové analyzátory mívají 24 až 35 kanálů, což je více než u logických analyzátorů. Tento počet plyne z požadavku na sledování alespoň 16 bitové adresovací sběrnice a 8 bitové datové sběrnice. Kapacita jednotlivých posuvných registrů bývá až 64 bitů, což je méně než u běžných logických analyzátorů. Způsoby synchronizace jsou obvykle stejné jako u logických analyzátorů.

Většina mikropočítačových analyzátorů používá pro zobrazování katodovou obrazovku. Zobrazování se provádí v osmičkové, nebo šestnáctkové soustavě, někdy pomocí mnemotechnického kódu. Některé typy mikroprocesorových analyzátorů zobrazují data ve sledu 1 - 0; použití časového diagramu není běžné. Některé nejjednodušší typy mikroprocesorových analyzátorů používají pro zobrazení polovodičové svíticí diody.

Nejčastější použití mikropočítačových analyzátorů je zkoušení uživatelských programů. Program může být zkoušen až do zvolené adresy (často do větvení programu), nebo může být krokován.